Steady-State Vibration of DFT Locomotive Cabs

Quinton Anthony Rowson

A Thesis Submitted in Fulfillment of the Requirements of the Degree

of

Masters of Mechanical Engineering

at the

University of Canterbury,

Christchurch, New Zealand.

2001.

To my Wife Fiona

Abstract

Back injuries and other illnesses for Tranz Rails Locomotive Engineers, is claimed to be attributed to vibration of locomotive cabs. The purpose of this research was to conduct a study in order to determine the complex motion of a DFT locomotive cab and to objectively and quantifiably measure the vibration present in the locomotive.

This thesis examines the ability of a six Degrees Of Freedom rigid body mathematical model of a DFT locomotive cab to determine the natural frequencies present. The goal was to use this model to provide information on different mount alternatives, in order to reduce vibration. The results of this model were inadequate for the selection of another elastomeric mounting alternative. While experimental test of the locomotive cab showed that the rigid body assumption was valid it was found that there are other continuous vibrations and non-linear effects that are also very important in determining cab vibration.

Auto-correlation results from acceleration measurements at the base of the mounts were overlaid with the cross-correlation of the acceleration measurements of the cab side of the mounts. These results showed that the current mounts have 100% transmissibility. In some cases and directions the mount was shown to actually amplify the vibrational input from the locomotive.

The amplitude of the steady-state vibration of the locomotive cab that was being transmitted was so great that it rendered the dynamic characteristics of the cab insignificant. From the experimental results, the elastomeric mounts were found to be incapable of attenuating the vibration. In addition, the vibration levels due to locomotive running are found to be well above comfort levels of various publications.

Acknowledgements

I would like to thank my wife, Fiona, for all her patience and proof reading.

There are also many other people that I would like to thank, who have made this thesis possible, in many ways, both academically and personally. First I would like to thank my supervisor, Ilanko, for his knowledgeable guidance, support, insight and encouragement, without which this thesis would not be possible.

I thank Harry M^{c}Callion and S Naguleswaran for their input, and guidance. Andy Cree for all the stupid questions that he let me ask him, and would sometimes answer, and for all his technical support regarding the computers, hardware etc.

I thank Tranz Rail Ltd. for providing me with the financial support for the first 15 months of my research. I would like to thank Bill Hudson from Tranz Rail for setting such a curious problem of such wide scope. Also for much of the material supplied at the initial stages of this project, and for giving me a taste of the industrial pressure that I will have to face for the rest of my engineering career, at the end of this project.

I am indebted to Geoff Chase for helping me finish my thesis, for his prompted reading of the final draft and comments of my thesis, and the holistic view that he was able to adopt.

At the personal level, I would like to thank my parents for instilling in me, that we are all capable of anything to which we set our minds. I thank Ann Ditcher for listening to my moans, and her advice. I would like to thank all my friends for their encouragement. Finally my office buddies Hamish Brice, Kathrin Winkelmann and Wang Kai for making my post-graduate time memorable.

Contents

List of Figures

List of Tables

Nomenclature

A	Transformation matrix for the $\underline{b}$ vector
$\underline{b}$	First measured coordinates vector
B	Transformation matrix for the $\underline{c}$ vector
c	Damping
c_{cr}	Critical damping
$\underline{c}$	Second measured coordinate vector
$\underline{\hat{c}}$	Estimation of second measured coordinate vector
C	Damping matrix
d_n	Longitudinal distance from the n^{th} mount to the centre of mass
D	Dissipation energy
e_n	vertical distance from the n^{th} mount to the centre of mass
f	Frequency
I_{ij}	Moment and Product inertia terms
k	Stiffness
K	Stiffness matrix
K*	Modified stiffness matrix
l_n	Lateral distance from the n^{th} mount to the centre of mass
m	Mass
M	Mass matrix
M*	Modified mass matrix
$N_{RMScomfort}$	Comfort index
q_k	Generalised coordinate
Qk	Generalised force corresponding to the generalized coordinate q_k.
Q	Dynamic Magnifier
R_{filt}	First order filter resistor
T	Kinetic energy
T_1	High time

T_2	Low time
V	Potential energy or volume
x	Longitudinal direction from the centre of mass
x_n	Longitudinal direction of the n^{th} mount
$\underline{x}$	Wanted coordinates vector for the centre of mass
y	Lateral direction from the centre of mass
y_n	Lateral direction of the n^{th} mount
z	vertical direction from the centre of mass
z_n	vertical direction of the n^{th} mount
θ_x	Roll, rotation about the longitudinal direction from the centre of mass
θ_y	Pitch, rotation about the lateral direction from the centre of mass
θ_z	Yaw, rotation about the vertical direction from the centre of mass
ρ	Density
ω	Driving Frequency
ω_n	Natural Frequency
ζ	Damping Ratio
ζ_n	Decay time constant of the n^{th} mode
?	Phase angle

Chapter 1

1 Introduction

The purpose of this research was to study vibrations that are present in the Diesel Freight Turbo (DFT) Locomotives leased by Tranz Rail Ltd. This research was carried out as a requirement of a Master of Mechanical Engineering Degree at the University of Canterbury, New Zealand. A photograph of a DFT locomotive can be seen in, Figure 1-1

Figure 1-1, Photo of DFT 7145 locomotive cab.

1.1 Company Background

Tranz Rail is a public company with a staff average of over 4,285 in the financial year of 1999. The total revenue for the 1999 financial year was $569.2 million dollars, while assets had a book value of $927.9 million dollars. This can be compared to the financial year of 1984[1] with a staff average of 19,747, a total revenue of $642 million

[1] Then known as the New Zealand Railways Corporation (NZR).

dollars and book value assets of $879 million dollars. By 1989 the average number of staff employed by New Zealand Railways Industry had fallen to 11,000. The main service that they provide is the transportation of freight throughout New Zealand. The three major registered share holders of Tranz Rail in August 1999 were New Zealand Central Securities Depository Limited (35.68%) Wisconsin Central International Inc (23.75%) and Pacific Rail Limited (18.71%).

As overviewed in the 1999 Annual Report, New Zealand Rail (NZR), which the government made a limited liability company in 1990, was privatised and sold for $400 million dollars in 1993. Its new owner was the consortium led by Wisconsin Central Transportation Corporation a publicly listed regional railroad in the United States. In 1995 New Zealand Rail became Tranz Rail, and was publicly listed in 1996 on the New Zealand Stock Exchange and the NASDAQ in the United States. In 1998 as Tranz Rail was finishing off the Middleton yard in Christchurch to create Tranz Rail's Transportation Centre the company was starting to feel the effects of the Asian economic crisis and a number of wide-ranging programs were started to cut costs.

Within New Zealand the two principal ways of moving freight are by rail or road. As stated in the 1999 Annual Report:

> *"Road transportation continues to enjoy a significant subsidy by not facing the full costs of the infrastructure it uses, nor the costs it imposes on society"*.

This subsidy has driven the Rail Industry of New Zealand[2] to become extremely competitive and more profitable. One of Tranz Rail's recent initiatives to increase profitability has been to maximise the utilization of equipment in service at all times.

Implementing a philosophy of having one hundred percent of locomotives in service at any one time has meant that there is no buffer to allow for regular maintenance of locomotives, equipment or unscheduled maintenance. This philosophy has led to the slow deterioration of aging locomotives and equipment and inevitably, a loss of

[2] Tranz Rail Ltd.

profits. It is speculated by the author that this approach may be a financially viable option due to the fact that the DFT locomotives are leased rather than owned. Tranz Rail continually tries to improve the efficiency of the company due to the pressures of the marketplace and from shareholders.

1.2 Project Development

In 1992 the first Diesel Freight (DF) locomotive was converted to a Diesel Freight Turbo (DFT). The new modification was 'trialed' for a period of 18 months. After this trial was completed the decision was made to convert the rest of the fleet, consisting of 30 DF's, to DFT's over a five-year period. A full account of the conversions that were necessary was supplied by Tranz Rail and is contained in Appendix A. Table 1-1 lists the specific conversions locomotive identification numbers.

Unfortunately, after the conversion to DFT's it was generally agreed by the drivers that the noise and vibration present in the DFT's was significantly greater than the original DF's. The situation escalated to the point where some drivers refused to drive some of the DFT locomotives. Initially, Tranz Rail performed considerable work to determine whether or not the vibrations present in the locomotive were causing, or contributing to, existing driver back pain or injuries. This investigation was motivated by a significant case of back pain/injury that Tranz Rail felt it needed to address.

Table 1-1, Rolling Stock sold to (and leased back from) GATX Corporation.

DFT7008	TURBO CONV ON DF 6260	DFT7173	TURBO CONV ON DF 6029
DFT7010	TURBO CONV ON DF 6300	DFT7186	TURBO CONV ON DF 6070
DFT7023	TURBO CONV ON DF 6248	DFT7199	TURBO CONV ON DF 6110
DFT7036	TURBO CONV ON DF 6317	DFT7200	TURBO CONV ON DF 6185
DFT7049	TURBO CONV ON DF 6219	DFT7213	TURBO CONV ON DF 6104
DFT7051	TURBO CONV ON DF 6231	DFT7226	TURBO CONV ON DF 6035
DFT7064	TURBO CONV ON DF 6254	DFT7239	TURBO CONV ON DF 6006 +BD
DFT7077	TURBO CONV ON DF 6225	DFT7241	TURBO CONV ON DF 6093 +BD
DFT7092	TURBO CONV ON DF 6283	DFT7254	TURBO CONV ON DF 6058
DFT7104	TURBO CONV ON DF 6277	DFT7267	TURBO CONV ON DF 6179
DFT7117	TURBO CONV ON DF 6156	DFT7282	TURBO CONV ON DF 6041
DFT7132	TURBO CONV ON DF 6191	DFT7295	TURBO CONV ON DF 6127
DFT7145	TURBO CONV ON DF 6012	DFT7307	TURBO CONV ON DF 6133
DFT7158	TURBO CONV ON DF 6202	DFT7322	TURBO CONV ON DF 6162
DFT7160	TURBO CONV ON DF 6087	DFT7335	TURBO CONV ON DF 6064

A limited literature review was carried out by the author indicated that Whole-Body-Vibration (WBV) can cause numerous symptoms including back pain and injury[3].

After significant consultation with all parties concerned it was decided that the vibration analysis problem was too complex for the institutions, such as Enviromedix, that were initially approached to find a solution. At that point, the University of Canterbury was approached and a contract was signed between the two bodies to the effect:

> *'The parties to this agreement plan to carry out research into Understanding and Minimising Vibration of DFT Locomotives.'*

1.3 Project Constraints

The main constraint that Tranz Rail placed on the project was that, to be of practical use to Tranz Rail, the solution to the vibration problem could not involve any change in the physical layout and construction of the cab or underframe of the DFT's. The reason for this constraint was the high financial cost the company might face in

implementing such findings. A second constraint placed on the project by Tranz Rail was that any parts used in a solution must be standard items that could be easily purchased from a catalogue. The reasoning was that Tranz Rail did not want to have any parts[4] specially designed, increasing the cost of alterations.

1.4 Research Objectives

In order to account for the Whole-Body-Vibration that the drivers are subjected to, the entire cab of the locomotive was considered rather than the drivers seat. Most studies that have been carried out in recent years have just considered the drivers seat.

There were two primary objectives defined for this research. The first objective was to develop and validate a mathematical model of the dynamics of the locomotive cab. Validation was to be found by comparing the damped natural frequencies calculated from the model with data obtained from experiments. The experimental damped natural frequencies could be measured by providing an impulse to the cab of the locomotive. A validated model could be used to investigate the effect of changing the stiffness and damping properties of the isolating mounts. By investigating how the system dynamics respond to the changes, before they are implemented, costly mistakes associated with inappropriate changes can be avoided.

The second objective was to design a data acquisition system, experimental testing procedures and a post-collection analysis procedure for the data. These systems would allow experimental data to be compared directly with the results generated using the mathematical model. This system could then be used to experimentally determine the significance of the vibrations present in the DFT cabs. Finally, this data could then be compared to accepted national and international standards on acceptable vibration exposure, such as ISO2631. This final step was to be done at the discretion of Tranz Rail after project completion.

---

[3] A review of the literature linking back pain and other adverse health effects is beyond the scope of this project.
[4] For example elastomeric mounts.

1.5 Literature Review

The development of the locomotive throughout the world is minimal in comparison to that of the preferred form of transport in New Zealand, the automobile. This result is mainly due to the enormous capital cost involved in the locomotive industry. In recent years the locomotive cab has had some attention with work carried out to reduce the vibration in older locomotives. Old locomotives were not designed with the idea of reducing Noise Vibration Harshness (NVH) in the locomotive cab for the comfort of the drivers. The cabs of these locomotives were designed purely to meet functional requirements. In part, this design approach was due to the lack of knowledge about available materials and the cost of mounting systems. These are the same type of locomotive used by Tranz Rail Ltd.

However, in recent years the rail industry has been able to transfer technology developed in other industries, such as the automotive industry, to improve NVH. Searches on the internet revealed the large range of products now available for the design of locomotives for low NVH levels to reduce WBV. However, these products do not overcome the difficulty of reducing the NVH levels in older locomotives as modifications can be very expensive. It is much easier to design a new locomotive to meet comfort, noise pollution, air pollution, and other requirements as opposed to redesigning existing, older locomotives.

Much of the academic literature concerning locomotive vibrations deals mainly with the wheel/rail interaction of the locomotives or the surrounding ground vibration of the locomotives [Iwnicki, 1988]. Until recently little research addressed the problem of vibrations that are present in the locomotive cab and the vibrations that the driver is subjected to.

Two Masters theses that addressed this problem are those of Venezia [1997] and Dhruna [1997] funded by General Electric Transportation Systems (GETS). Venezia [1997] and Dhruna [1997] used a full-scale model of the locomotive cab supplied by

GETS to emulate actual vibration input to the cab structure. Correlating baseline measurements from laboratory vibration with field data, experiments were performed to evaluate vibration contribution to different parts of the locomotive cab. The cab floor and roof proved to have the greatest vibrations. Several solutions were implemented with various results, but none of the solutions significantly reduced the noise present in the cab. A mathematical model of the locomotive cab was also developed in MATLAB but did not produce accurate results. Both authors recommended that further research should be undertaken to improve the model.

A publication that provided a benchmark for discrete system modelling was that of Grootenhuis [1965]. This research considered a rigid body with six degrees of freedom, and three axes of asymmetry supported on four springs. The model allowed for the lateral as well as the longitudinal stiffness of the springs. However, this model did not take damping into account. Coupled motion in all directions occurred even when the centre-of-mass was at the geometric centre of the rigid body, apart from the vibration in the vertical axis (which coincides as the longitudinal direction of the four springs) and rotation vibration about the vertical axis. Multiple springs were allowed for by adding terms to the expression of the potential energy of the springs. Product inertia terms were also accounted for when the body's centre-of-mass was off-set, by the expression for the kinetic energy.

A model of a rectangular frame replaced the rigid body, with all of the mass concentrated at the eight corners for the purpose of the simulation. The matrix solution was performed in dimensionless parameters, which made the study of the effects of the eigenvalues and eigenvectors due to the change in the position of the centre of mass easier. Proportions between stiffness ratios of the lateral to longitudinal stiffness of the springs was the only factor remaining in the analysis. It was discovered that small amounts of off-set of the centre-of-mass from the geometric centre minutely changed the dynamic behaviour of the system.

A paper by Li & Yam [2000] considered a massive rotating machine resiliently supported by an elastic structure using a component synthesis method for vibration isolation design. The system is considered as two bodies, consisting of the machine

and supporting structure connected by resilient elements. The stiffness matrix for the connecting resilient elements of any configuration and location can be obtained. The equations of motion and the vibrational characteristics of the synthesized system may then be obtained. Generally, to reduce vibration transmissibility, springs with lower stiffness are used between the frame and the machine. In such a case, the natural frequencies of the coupled system can be divided into a low band and a high band. Six degrees of freedom were considered in the paper for the resilient supported body. The paper states that as the machine size is increased, the stiffness of the frame will decrease, and the natural frequencies of the higher band will decrease. As the lower frequency of the high band approaches the operating frequency of the rotating machine, resonance of the support structure is induced. The motion of the machine was described by three translational displacements of the centre of mass of the machine, with three rotational displacements about the inertial axes. The characteristics of the elastic frame were calculated using a finite element method. The Modal Synthesis Method was adopted to derive the general equations of motion for the coupled system and results of the numerical calculations indicated good agreement with experimental results.

The purpose of a study by Ahmadian [1995], was to model and simulate the vibration that occurred in the cab of a locomotive. This work was attempted using a method of mechanical impedance of an element, which is defined as the ratio of the driving force acting on the element to the resulting velocity, as defined by Meirovitch [1986]. This ratio was then used to select the required mounts for the cab.

Wilde [1980] overviews important information regarding vibration frequencies affecting humans. The paper also overviews all of the several topics in relation to whole body vibration exposure.

1.6 Outline & Overview

This thesis demonstrates the design of the mathematical model and of an experimental data acquisition system used for investigating the mechanical vibrations of a DFT

locomotive cab. The theoretical results from the simulation and the experimental results of the locomotive for the damped natural frequencies are then compared and discussed. Chapter 2 discusses the purpose of the mathematical model, the assumptions that were made and the derivation of the model, concluding with some of the results. Chapter 3 discusses the requirements and justification of the experimental data acquisition, hardware and software used to obtain the natural frequencies of the locomotive cab. Chapter 4 discusses the methods and purpose for the experimental testing, as well as presenting the results. Chapter 5 discusses and analyses the results from the mathematical model in Chapter 2 and the experimental results of Chapter 4. Chapter 6 presents the research conclusions and ends with recommendations for further research.

Chapter 2

2 Theory and Mathematical Modelling

The purpose of simulation is to provide a means of investigating how the system dynamics of the locomotive cab respond to changes in the cab mounting before they are implemented. This approach reduces the man-hours and material involved in changing mounts and making alterations that would be necessary if all possible solutions were trialled.

2.1 Isolation Selection

Due to the constraint that the general physical construction of the DFT locomotive not be changed, the options that could be used to solve vibrational issues in the locomotive cab were greatly reduced. Venezia [1997] provides full background on the different types of mounting systems that could be used for isolating the locomotive cab including the advantages and disadvantages of each. The reader should refer to the relevant literature found in Appendix B.

The main designs considered for isolating such a structure are:

- Elastomeric Mounts.
- Airbag Springs.
- Fluid Mounts.
- Active Mounts.
- Coil Springs.

Elastomeric mounts are currently in use on Tranz Rail's DFT locomotives. After investigating each type of mount, it appeared that the most appropriate type of mount

to isolate the DFT cabs may be an optimised elastomeric mount. This type of mount exhibits suitable stiffness and damping properties, and would satisfy Tranz Rail's constraint that no significant changes could be made to the physical configuration of the DFT cab and under-frame.

2.2 Elastomeric Mounts

The elastomeric mounts used on Tranz Rail's DFT fleet provide lateral and rotational stiffness and damping on and around both the axial and radial directions. The configuration of the mount when located on the under-frame of the DFT locomotive is shown in Figure 2-1. A detailed schematic of the generic mounting configuration of the mount is shown in Figure 2-2. The elastomeric mount currently used on Tranz Rail's DFT is a Metalastik Ltd product. More details on the mounts are given in Metalastik catalogue and an excerpt from the catalogue can be viewed in Appendix C.

Figure 2-1, Picture of the location of a DFT mount attached to the underframe.

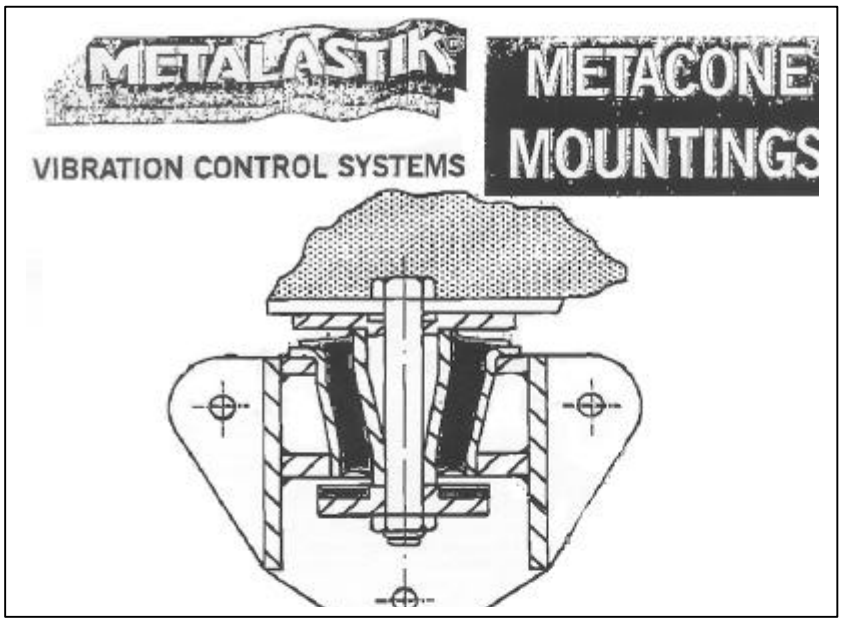


Figure 2-2, Schematic of the mount used on DFT's

2.3 Non-Linear Behaviour of the Mounts

Due to the properties of the elastomer from which the mount is fabricated, the mount exhibits internal damping. Therefore, in the case of forced vibration the damping present is dependent on the frequency of vibration. By calculating an equivalent viscous damping coefficient the equations of motion can be linearised and the problem greatly simplified.

To calculate the equivalent viscous damping coefficient an appropriate dynamic magnifier must be determined. By calculating the driving frequency of a locomotive, Metalastik were able to supply the appropriate dynamic magnifier value. The calculation of the equivalent viscous damping coefficient is shown in Equation (2-1).

$$Q = \frac{1}{\sqrt{\left(1 - \frac{w^2}{w_n^2}\right)^2 + \left(2z\,\frac{w}{w_n}\right)^2}}$$

Equation (2-1)

Where: Q = Dynamic magnifier

w = Driving frequency of the system

$w_n = \sqrt{k/m}$ = Natural frequency of the system

$z = c / c_{cr}$ = The damping ratio

c = The damping of the system

$c_{cr} = 2\sqrt{km}$ = The critical damping of the system

Algebraic manipulation and use of the fact that the maximum amplitude ratio occurs when the ratio of the driving frequency to the natural frequency is the specific value shown in Equation (2-2) leads to the result in Equation (2-3).

$$\frac{w}{w_n} = \sqrt{1-2z^2}$$

Equation (2-2)

$$Q^2 = \frac{1}{-4z^4 + 4z^2}$$

Equation (2-3)

Using Equation (2-3) it is possible to obtain a polynomial for the damping coefficient, c, as shown in Equation (2-4).

$$\therefore \quad c^4\left(\frac{-4Q^2}{c_{cr}^4}\right) + c^2\left(\frac{4Q^2}{c_{cr}^2}\right) - 1 = 0$$

Equation (2-4)

Equation (2-4) can be solved, leaving the following result.

$$\therefore \quad c = \pm\sqrt{\frac{-\left(\frac{4Q^2}{c_{cr}{}^2}\right) \pm \sqrt{\left(\frac{4Q^2}{c_{cr}{}^2}\right)^2 - 4\left(\frac{-4Q^2}{c_{cr}{}^2}\right)(-1)}}{2\left(\frac{-4Q^2}{c_{cr}{}^4}\right)}}$$

Equation (2-5)

The only result needed is the positive real solution so the other solutions to Equation (2-5) can be discarded. This solution can be considered representative of the inherent viscous damping of the mount. The solution assumes small motion, however with approximately 6 mm maximum static deflection, the assumption of viscous damping is safe for these much larger mounts.

The calculation of the damping coefficient, c, for this mount can be seen in Appendix D within the MATLAB code. The calculated critical damping, c_{cr}, in the axial and radial directions of the mount are 80515 Ns/m and 150660 Ns/m respectively. From the formulation of Equation (2-5) the calculated axial and radial damping, c, of the mounts were 3099 and 5799 Ns/m respectively. This data gives a calculated damping ratio, z , of the mounts of 0.0385 in both the axial and radial directions.

2.4 Simulation Requirements and Assumptions

The general requirement of the simulation was that it had to depict the physical situation that occurs within the DFT locomotive cabs. After riding in the locomotive cab and viewing the physical structure of the locomotive cab on many occasions, it was decided that the rigidity of the locomotive cab was sufficient enough to allow the cab to be modelled as a rigid body.

To simulate the vibration of the locomotive to any degree, it is necessary to use all six degrees of freedom. The six degrees of freedom include the three translational directions, and three rotational directions. The three translation directions are

longitudinal, lateral and vertical or x, y, and z respectively, and the three rotational directions are roll, pitch and yaw or $?_x$, $?_y$, and $?_z$ respectively.

2.5 SolidWorks Model of the Locomotive Cab

Before the eigenvectors and eigenvalues can be calculated it was necessary to have an accurate calculation for the mass of the locomotive cab as well as mass moments and products of inertia.

A SolidWorks model was created by referring to the cab general arrangement drawings that were available for the DFT, and by numerous measurements of the cab dimensions and wall thicknesses. The model was then developed by creating all of the sides of the model, such as the floor, front, back, sides and roof.

The mass obtained was 3294kgs.

The coordinate of the centre of mass is[1]:

$$\begin{Bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{Bmatrix} = \begin{Bmatrix} 1.06537 \\ 0.76157 \\ 1.62526 \end{Bmatrix} (metres)$$

The moments of inertia taken at the centre of mass and aligned with the coordinate system above are:

I_{xx}=3011.92308 kgm^2,	I_{xy}=-80.60274 kgm^2,	I_{xz}=-154.05403 kgm^2
I_{yx}=-80.60274 kgm^2,	I_{yy}=3069.80909 kgm^2,	I_{yz}=78.57755 kgm^2
I_{zx}=-154.05403 kgm^2,	I_{zy}=78.57755 kgm^2,	I_{zz}=2106.52262 kgm^2

[1] Note that the coordinate system of the SolidWorks model differs from the coordinate system of the simulation. The solid works model used the right-handed Cartesian coordinate system, where z is the lateral direction and y is the vertical direction. However to conform to the ISO standard for vibration of a human in vibrational environment the right-hand Cartesian coordinate system was used where, z is the vertical axis, and y is the lateral axis. In both cases x is the longitudinal axis.

Figure 2-3 shows an isometric view of the solid model that SolidWorks produced as well as the coordinate system of the model with the prescribed directions of the coordinates labelled x, y, and z. The green coordinate system is the actual coordinate system and origin for the SolidWorks model. The red coordinate system indicates the principal coordinates for the centre of mass of the model.

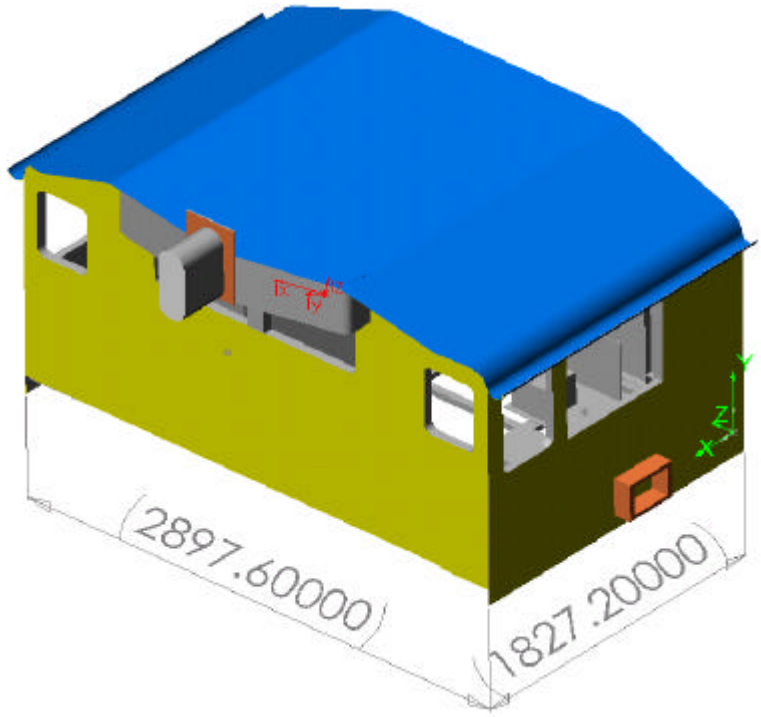


Figure 2-3, SolidWorks Model of DFT cab.

2.6 Development of the Six Degree of Freedom Model for Simulation

A diagram of the model used for the development of the equations of motion and the development of all the conventions used can be seen Figure 2-4. All of the DFT locomotive cabs have the same physical construction, however variables from the diagram were used for each dimension to allow for unexpected differences in the construction of the cabs. The construction of the cab has four elastomeric mounts,

which are indicated by the locations marked 1, 2, 3 and 4. As indicated in section 2.2, stiffness and damping in all of the translational directions of the mount are necessary. The model therefore allows the use of stiffness and damping coefficients in the translational directions, (x, y, and z) at all four mounts.

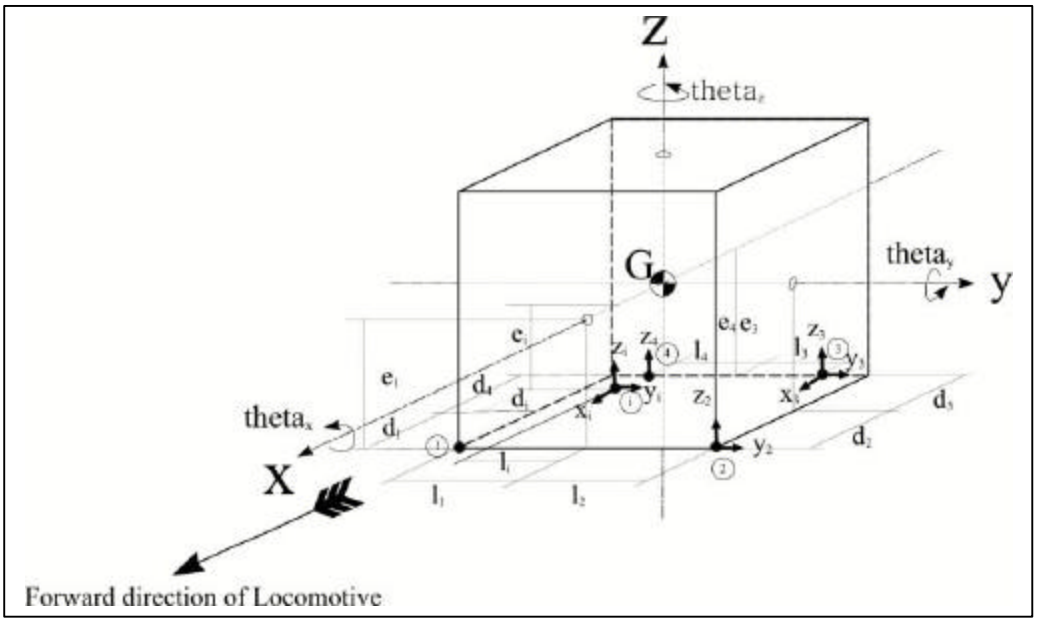


Figure 2-4, Model of six degrees of freedom.

The 6 major coordinates shown in Figure 2-4 that are required to make the coordinate transformation of x, y, z, $?_x$, $?_y$, and $?_z$ are shown with large normal arrows. A transformed set of coordinates is also shown with labelling next to the bold arrows of y_2, z_2, x_3, y_3, z_3, z_4, x_i, y_i, and z_i. For example, 'x' indicates the direction of the coordinate that was measured from the mounted accelerometer, and the subscript number or letter that follows that vectorial direction indicates the position of the measurement from the mounted accelerometer.

The subscripts indicate the position of the mount. For example, 'dot' with the four encircled beside it, indicates the position of the fourth mount that suspends the cab of the locomotive from the underframe of the locomotive. The exact location of the fourth mount is indicated by the distance e_1 in the vertical axis, from the centre of mass. The lateral distance from the centre of mass to mount 4 is indicated by l_4. The

longitudinal distance from the centre of mass to mount 4 is indicated by d_4. The subscript "i" indicates the position of the driver's seat within the locomotive cab. For the purpose of the rigid body test, the coordinates x_1, y_1, z_1, x_2 , x_4, and y_4 were also collected.

For small displacement models the assumption has been made that sin(?)=?, and cos(?)=1-(?^{2})/2. An arbitrary displacement in each direction is considered, namely, x, y, z, ?$_x$, ?$_y$, and ?$_z$. To use Lagrange's equations in Equation (2-6) the potential energy stored in the mounts must be formulated as shown in Equation (2-7). The Lagrange's equation of motion is [McCallion, 1973]:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_k}\right) - \frac{\partial T}{\partial q_k} + \frac{\partial D}{\partial \dot{q}_k} + \frac{\partial V}{\partial q_k} = Q_k$$

Equation (2-6)

Where T is the kinetic energy of the system, V is the potential energy of the system, D is the dissipation energy due to factors such as damping, and Q_k is the generalised force corresponding to the generalised coordinate q_k. In the present analysis, the six coordinates corresponding to the translation and rotation of the centre of the mass (x, y, z, ?$_x$, ?$_y$, and ?$_z$) are the generalised coordinates.

The potential energy consists of the following terms:

$$V_{k_1} = \frac{1}{2}\left\{k_1\left(z - l_1 q_x - d_1 q_y\right)^2 + k_1\left(z + l_2 q_x - d_2 q_y\right)^2 + k_1\left(z + l_3 q_x + d_3 q_y\right)^2 + k_1\left(z - l_4 q_x + d_4 q_y\right)^2\right\}$$

Equation (2-7)

Equation (2-7) is the potential energy due to the relative axial displacement of the mounts, represented by the subscript 1, which is in the z-direction of Figure 2-4. Similarly the potential energy associated with the radial displacement of the mounts are given by Equation (2-8) and Equation (2-9), for the y-direction and the x-direction respectively.

$$V_{k_2} = \frac{1}{2}\left\{k_2\left(y + e_1 q_x + d_1 q_z\right)^2 + k_2\left(-y - e_2 q_x - d_2 q_z\right)^2 + k_2\left(-y - e_3 q_x + d_3 q_z\right)^2 + k_2\left(y + e_4 q_x - d_4 q_z\right)^2\right\}$$

Equation (2-8)

$$V_{k_3} = \frac{1}{2}\left\{k_3\left(-x - l_1 q_z + e_1 q_y\right)^2 + k_3\left(-x + l_2 q_z + e_2 q_y\right)^2 + k_3\left(x - l_3 q_z - e_3 q_y\right)^2 + k_3\left(x + l_4 q_z - e_4 q_y\right)^2\right\}$$

Equation (2-9)

The dissipation energy, D, in the mounts is given in Equation (2-10) to Equation (2-12).

$$D_{c_1} = \frac{1}{2}\left\{c_1\left(\dot{z} - l_1 \dot{q}_x - d_1 \dot{q}_y\right)^2 + c_1\left(\dot{z} + l_2 \dot{q}_x - d_2 \dot{q}_y\right)^2 + c_1\left(\dot{z} + l_3 \dot{q}_x + d_3 \dot{q}_y\right)^2 + c_1\left(\dot{z} - l_4 \dot{q}_x + d_4 \dot{q}_y\right)^2\right\}$$

Equation (2-10)

Equation (2-10) is the axial direction of the mount, represented by the subscript 1, which is in the z-direction of Figure 2-4.

$$D_{c_2} = \frac{1}{2}\left\{c_2\left(\dot{y} + e_1 \dot{q}_x + d_1 \dot{q}_z\right)^2 + c_2\left(-\dot{y} - e_2 \dot{q}_x - d_2 \dot{q}_z\right)^2 + c_2\left(-\dot{y} - e_3 \dot{q}_x + d_3 \dot{q}_z\right)^2 + c_2\left(\dot{y} + e_4 \dot{q}_x - d_4 \dot{q}_z\right)^2\right\}$$

Equation (2-11)

Equation (2-11) is for the radial direction of the mount, represented by the subscript 2, which is the y-direction of Figure 2-4.

$$D_{c_3} = \frac{1}{2}\left\{c_3\left(-\dot{x} - l_1 \dot{q}_z + e_1 \dot{q}_y\right)^2 + c_3\left(-\dot{x} + l_2 \dot{q}_z + e_2 \dot{q}_y\right)^2 + c_3\left(\dot{x} - l_3 \dot{q}_z - e_3 \dot{q}_y\right)^2 + c_3\left(\dot{x} + l_4 \dot{q}_z - e_4 \dot{q}_y\right)^2\right\}$$

Equation (2-12)

Equation (2-12) is for the radial direction of the mount, represented by the subscript 3, which is the x-direction of Figure 2-4.

The total kinetic energy, T, of the body is given in Equation (2-13) and simplified in Equation (2-14)

$$T = \frac{1}{2}\left\{m\dot{x}^2 + m\dot{y}^2 + m\dot{z}^2 + I_{xx}\dot{q}_x^{\,2} + I_{yy}\dot{q}_y^{\,2} + I_{zz}\dot{q}_z^{\,2} - I_{xy}\dot{q}_x\dot{q}_y - I_{xz}\dot{q}_x\dot{q}_z - I_{yx}\dot{q}_y\dot{q}_x - I_{yz}\dot{q}_y\dot{q}_z - I_{zx}\dot{q}_z\dot{q}_x - I_{zy}\dot{q}_z\dot{q}_x\right\}$$

Equation (2-13)

$$T = \frac{1}{2}\left\{m\dot{x}^2 + m\dot{y}^2 + m\dot{z}^2 + I_{xx}\dot{q}_x^{\,2} + I_{yy}\dot{q}_y^{\,2} + I_{zz}\dot{q}_z^{\,2} - 2I_{xy}\dot{q}_x\dot{q}_y - 2I_{xz}\dot{q}_x\dot{q}_z - 2I_{yz}\dot{q}_y\dot{q}_z\right\}$$

Equation (2-14)

Substituting Equation (2-7) through Equation (2-14) into Equation (2-6) the equations of motion can be derived for each of the 6 directions in Equation (2-15) through Equation (2-20).

$$\begin{aligned} &m\ddot{x} + k_1(x + l_1q_z - e_1q_y) + k_1(x - l_2q_z - e_2q_y) + k_1(x - l_3q_z - e_3q_y) + k_1(x + l_4q_z - e_4q_y) \\ &\quad + c_1(\dot{x} + l_1\dot{q}_z - e_1\dot{q}_y) + c_1(\dot{x} - l_2\dot{q}_z - e_2\dot{q}_y) + c_1(\dot{x} - l_3\dot{q}_z - e_3\dot{q}_y) + c_1(\dot{x} + l_4\dot{q}_z - e_4\dot{q}_y) = 0 \end{aligned}$$

Equation (2-15)

$$\begin{aligned} &m\ddot{y} + k_2(y + e_1q_x + d_1q_z) + k_2(y + e_2q_x + d_2q_z) + k_2(y + e_3q_x - d_3q_z) + k_2(y + e_4q_x - d_4q_z) \\ &\quad + c_2(\dot{y} + e_1\dot{q}_x + d_1\dot{q}_z) + c_2(\dot{y} + e_2\dot{q}_x + d_2\dot{q}_z) + c_2(\dot{y} + e_3\dot{q}_x - d_3\dot{q}_z) + c_2(\dot{y} + e_4\dot{q}_x - d_4\dot{q}_z) = 0 \end{aligned}$$

Equation (2-16)

$$\begin{aligned} &m\ddot{z} + k_1(z - l_1q_x - d_1q_y) + k_1(z + l_2q_x - d_2q_y) + k_1(z + l_3q_x + d_3q_y) + k_1(z - l_4q_x + d_4q_y) \\ &\quad + c_1(\dot{z} - l_1\dot{q}_x - d_1\dot{q}_y) + c_1(\dot{z} + l_2\dot{q}_x - d_2\dot{q}_y) + c_1(\dot{z} + l_3\dot{q}_x + d_3\dot{q}_y) + c_1(\dot{z} - l_4\dot{q}_x + d_4\dot{q}_y) = 0 \end{aligned}$$

Equation (2-17)

$$I_{xx}\ddot{q}_x - I_{xy}\ddot{q}_y - I_{xz}\ddot{q}_z + l_1k_1(-z + l_1q_x + d_1q_y) + l_2k_1(z + l_2q_x - d_2q_y) + l_2k_1(z + l_2q_x + d_2q_y) + l_2k_1(-z + l_2q_x d_2q_y)$$
$$+ e_1k_2(y + e_1q_x + d_1q_z) + e_2k_2(y + e_2q_x + d_2q_z) + e_2k_2(y + e_2q_x - d_2q_z) + e_2k_2(y + e_2q_x - d_2q_z)$$
$$+ l_1c_1(-\dot{z} + l_1\dot{q}_x + d_1\dot{q}_y) + l_2c_1(\dot{z} + l_2\dot{q}_x - d_2\dot{q}_y) + l_2c_1(\dot{z} + l_2\dot{q}_x + d_2\dot{q}_y) + l_2c_1(-\dot{z} + l_2\dot{q}_x d_2\dot{q}_y)$$
$$+ e_1c_2(\dot{y} + e_1\dot{q}_x + d_1\dot{q}_z) + e_2c_2(\dot{y} + e_2\dot{q}_x + d_2\dot{q}_z) + e_2c_2(\dot{y} + e_2\dot{q}_x - d_2\dot{q}_z) + e_2c_2(\dot{y} + e_2\dot{q}_x - d_2\dot{q}_z) = 0$$

Equation (2-18)

$$I_{yy}\ddot{q}_y - I_{yx}\ddot{q}_x - I_{yz}\ddot{q}_z + d_1k_1(-z + l_1q_x + d_1q_y) + d_2k_1(-z - l_2q_x + d_2q_y) + d_3k_1(z + l_3q_x + d_3q_y) + d_4k_1(z - l_4q_x + d_4q_y)$$
$$+ e_1k_3(-x - l_1q_z + e_1q_y) + e_2k_3(-x + l_2q_z + e_2q_y) + e_3k_3(-x + l_3q_z + e_3q_y) + e_4k_3(-x - l_4q_z + e_4q_y)$$
$$+ d_1c_1(-\dot{z} + l_1\dot{q}_x + d_1\dot{q}_y) + d_2c_1(-\dot{z} - l_2\dot{q}_x + d_2\dot{q}_y) + d_3c_1(\dot{z} + l_3\dot{q}_x + d_3\dot{q}_y) + d_4c_1(\dot{z} - l_4\dot{q}_x + d_4\dot{q}_y)$$
$$+ e_1c_3(-\dot{x} - l_1\dot{q}_z + e_1\dot{q}_y) + e_2c_3(-\dot{x} + l_2\dot{q}_z + e_2\dot{q}_y) + e_3c_3(-\dot{x} + l_3\dot{q}_z + e_3\dot{q}_y) + e_4c_3(-\dot{x} - l_4\dot{q}_z + e_4\dot{q}_y) = 0$$

Equation (2-19)

$$I_{zz}\ddot{q}_z - I_{zx}\ddot{q}_x - I_{zy}\ddot{q}_y + d_1k_2(y + e_1q_x + d_1q_z) + d_2k_2(y + e_2q_x + d_2q_z) + d_2k_2(-y - e_2q_x + d_2q_z) + d_2k_2(-y - e_2q_x + d_2q_z)$$
$$+ l_1k_3(x + l_1q_z - e_1q_y) + l_1k_3(-x + l_1q_z + e_1q_y) + l_1k_3(-x + l_1q_z + e_1q_y) + l_1k_3(x + l_1q_z - e_1q_y)$$
$$+ d_1c_2(\dot{y} + e_1\dot{q}_x + d_1\dot{q}_z) + d_2c_2(\dot{y} + e_2\dot{q}_x + d_2\dot{q}_z) + d_2c_2(-\dot{y} - e_2\dot{q}_x + d_2\dot{q}_z) + d_2c_2(-\dot{y} - e_2\dot{q}_x + d_2\dot{q}_z)$$
$$+ l_1c_3(\dot{x} + l_1\dot{q}_z - e_1\dot{q}_y) + l_1c_3(-\dot{x} + l_1\dot{q}_z + e_1\dot{q}_y) + l_1c_3(-\dot{x} + l_1\dot{q}_z + e_1\dot{q}_y) + l_1c_3(\dot{x} + l_1\dot{q}_z - e_1\dot{q}_y) = 0$$

Equation (2-20)

This energy formulation was compared to a paper by Grootenhuis[1965] and the terms were found to be the same. In addition, the dissipative terms have been considered in the energy formulation. These equations may be expressed in matrix form in terms of: the mass matrix M, damping matrix C, and stiffness matrix K, can be formed as shown in Equation (2-21) to Equation (2-23).

$$M = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & -I_{xy} & -I_{xz} \\ 0 & 0 & 0 & -I_{yx} & I_{yy} & -I_{yz} \\ 0 & 0 & 0 & -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix}$$

Equation (2-21)

$$C = \begin{bmatrix} \overbrace{4c_3}^{\dot{x}} & \overbrace{0}^{\dot{y}} & \overbrace{0}^{\dot{z}} & \overbrace{0}^{\dot{q}_x} & \overbrace{c_3(-e_1-e_2-e_3-e_4)}^{\dot{q}_y} & \overbrace{c_3(l_1-l_2-l_3+l_4)}^{\dot{q}_z} \\ 0 & 4c_2 & 0 & c_2(e_1+e_2+e_3+e_4) & 0 & c_2(d_1+d_2-d_3-d_4) \\ 0 & 0 & 4c_1 & c_1(-l_1+l_2+l_3-l_4) & c_1(-d_1-d_2+d_3+d_4) & 0 \\ 0 & c_2(e_1+e_2+e_3+e_4) & c_1(-l_1+l_2+l_3-l_4) & c_1(l_1^2+l_2^2+l_3^2+l_4^2)+c_2(e_1^2+e_2^2+e_3^2+e_4^2) & c_1(-l_1d_1+l_2d_2+l_3d_3-l_4d_4) & c_2(e_1d_1+e_2d_2-e_3d_3-e_4d_4) \\ c_3(-e_1-e_2-e_3-e_4) & 0 & c_1(-d_1-d_2+d_3+d_4) & c_1(l_1d_1-l_2d_2+l_3d_3-l_4d_4) & c_1(d_1^2+d_2^2+d_3^2+d_4^2)+c_3(e_1^2+e_2^2+e_3^2+e_4^2) & c_3(-e_1l_1+e_2l_2+e_3l_3-e_4l_4) \\ c_3(l_1-l_2-l_3+l_4) & c_2(d_1+d_2-d_3-d_4) & 0 & c_2(e_1d_1+e_2d_2-e_3d_3-e_4d_4) & c_3(-e_1l_1+e_2l_2+e_3l_3-e_4l_4) & c_2(d_1^2+d_2^2+d_3^2+d_4^2)+c_3(l_1^2+l_2^2+l_3^2+l_4^2) \end{bmatrix} \begin{matrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{q}_x \\ \dot{q}_y \\ \dot{q}_z \end{matrix}$$

Equation (2-22)

$$K=\begin{bmatrix} \overbrace{4k_3}^{x} & \overbrace{0}^{y} & \overbrace{0}^{z} & \overbrace{0}^{q_x} & \overbrace{k_3(-e_1-e_2-e_3-e_4)}^{q_y} & \overbrace{k_3(l_1-l_2-l_3+l_4)}^{q_z} & x \\ 0 & 4k_2 & 0 & k_2(e_1+e_2+e_3+e_4) & 0 & k_2(d_1+d_2-d_3-d_4) & y \\ 0 & 0 & 4k_1 & k_1(-l_1+l_2+l_3-l_4) & k_1(-d_1-d_2+d_3+d_4) & 0 & z \\ 0 & k_2(e_1+e_2+e_3+e_4) & k_1(-l_1+l_2+l_3-l_4) & k_1(l_1^2+l_2^2+l_3^2+l_4^2)+k_2(e_1^2+e_2^2+e_3^2+e_4^2) & k_1(-l_1d_1+l_2d_2+l_3d_3-l_4d_4) & k_2(e_1d_1+e_2d_2-e_3d_3-e_4d_4) & q_x \\ k_3(-e_1-e_2-e_3-e_4) & 0 & k_1(-d_1-d_2+d_3+d_4) & k_1(l_1d_1-l_2d_2+l_3d_3-l_4d_4) & k_1(d_1^2+d_2^2+d_3^2+d_4^2)+k_3(e_1^2+e_2^2+e_3^2+e_4^2) & k_3(-e_1l_1+e_2l_2+e_3l_3-e_4l_4) & q_y \\ k_3(l_1-l_2-l_3+l_4) & k_2(d_1+d_2-d_3-d_4) & 0 & k_2(e_1d_1+e_2d_2-e_3d_3-e_4d_4) & k_3(-e_1l_1+e_2l_2+e_3l_3-e_4l_4) & k_2(d_1^2+d_2^2+d_3^2+d_4^2)+k_3(l_1^2+l_2^2+l_3^2+l_4^2) & q_z \end{bmatrix}$$

Equation (2-23)

The moment and products of inertia are formulated as shown in Equation (2-24).

$$I_{xx} = \int r\left(\bar{y}^2 + \bar{z}^2\right)dn \quad , \qquad I_{\bar{x}\,\bar{y}} = \int r\bar{x}\bar{y}dn$$

Equation (2-24)

MATLAB was used for the calculation of the natural frequencies using the standard damped eigenvalue problem of Equation (2-25).

$$\left(K^* - w^2 M^*\right)\mathbf{q} = 0$$

Equation (2-25)

where:

$$K^* = \begin{bmatrix} -M & 0 \\ 0 & K \end{bmatrix}$$

Equation (2-26)

$$M^* = \begin{bmatrix} 0 & M \\ M & C \end{bmatrix}$$

Equation (2-27)

From these equations the eigenvalues (natural frequencies), w, and the eigenvectors (mode shapes), $\mathbf{q}$, can be calculated. There are several papers that also take this approach to modelling a locomotive cab [Butsuen, 1987; Grootenhuis, 1965; Li, 2000].

The natural frequencies (the imaginary part of the eigenvalue, ω_n, multiplied by 2π) and damping coefficient, $?_n$, for the unmodified, existing DFT cabs are shown in Table 2-1.

Table 2-1, Results of the Mathematical Model using the Current Mount Properties used of the DFT Locomotive cabs.

Mode Number (i)	Frequency, f_i (Hz)	Damping Coefficient ($?_i$)
1	5.4	0.87176
2	6.7	1.3401
3	8.2	2.0278
4	18.8	6.3461
5	20.1	7.3807
6	28.3	13.364

The results shown in Table 2-1 arrise from the fact that when you have a damped system the eigenvalues for the system comprise of a real part and an imaginary part. The real part indicated the damping constant of the free vibration. The imaginary part of the eigenvalues is proportional to the natural frequencies of the system by 2π. This does not occur in an undamped system, and the eigenvalues comprise of only real numbers, which are proportional to the natural frequencies of the system by 2π.

For each mode, it was important to detect the relative magnitudes of each element of the mode vector. In order to do this a method of correlating the rotational vector components $?_x$, $?_y$, and $?_z$ to the translational components x, ,y, and z was used. Then the most dominant coordinates could be detected for each mode. In order to accomplish this task the magnitude of the rotational coordinates $?_x$, $?_y$, and $?_z$ are multiplied by the radial, distance about the rotational axis concerned, from the centre of mass to the corner of the cab where the mount is situated. The magnitudes of the eigenvector displacements for each mode could then be directly compared as shown in Table 2-2 to Table 2-7.

Table 2-2, Mode 1 for Frequency, f_1=5.4(Hz)

Magnitude (Normalised) x10^{-4}	7.6475	90.466	191.6	206.58	1.4467	8.6588
Phase Angle (deg)	72.871	72.922	106.78	106.99	105.28	107.24
Order	(y dir)	(z dir)	(x dir)	(?$_y$ dir)	(?$_z$ dir)	(?$_x$ dir)

Table 2-3, Mode 2 for Frequency, f_2=6.7(Hz).

Magnitude (Normalised) x10^{-4}	155.56	88.644	20.82	19.716	3.4456	152.03
Phase Angle (deg)	25.712	26.497	26.538	27.085	24.013	153.84
Order	(y dir)	(z dir)	(x dir)	(?$_y$ dir)	(?$_z$ dir)	(?$_x$ dir)

Table 2-4, Mode 3 for Frequency, f_3=8.2(Hz)

Magnitude (Normalised) x10^{-4}	59.198	169.58	42.759	36.858	0.9527	50.475
Phase Angle (deg)	4.8567	175.48	175.6	174.67	3.1073	174.19
Order	(y dir)	(z dir)	(x dir)	(?$_y$ dir)	(?$_z$ dir)	(?$_x$ dir)

Table 2-5, Mode 4 for Frequency, f_4=18.8(Hz)

Magnitude (Normalised) x10^{-4}	33.072	0.7517	45.607	44.522	26.74	35.421
Phase Angle (deg)	143.17	25.288	145.3	33.775	145.34	144.46
Order	(y dir)	(z dir)	(x dir)	(?$_y$ dir)	(?$_z$ dir)	(?$_x$ dir)

Table 2-6, Mode 5 for Frequency, f_5=20.1(Hz)

Magnitude (Normalised) x10^{-4}	38.577	2.9447	35.146	39.213	3.032	44.6
Phase Angle (deg)	157.08	163.03	24.436	156.69	165.72	158.49
Order	(y dir)	(z dir)	(x dir)	(?$_y$ dir)	(?$_z$ dir)	(?$_x$ dir)

Table 2-7, Mode 6 for Frequency, f_6=28.3(Hz)

Magnitude (Normalised) x10^{-4}	6.6025	3.27653	5.2068	6.0316	54.709	7.7852
Phase Angle (deg)	77.294	45.492	77.521	102.91	102.33	76.432
Order	(y dir)	(z dir)	(x dir)	(?$_y$ dir)	(?$_z$ dir)	(?$_x$ dir)

The mode shapes shown in Table 2-2 to Table 2-7 are the modulus of the modes. It can be seen that there is a definite upper and lower band on natural frequencies as also pointed out in Li [2000], as the cab is small in comparison to the rest of the locomotive.

These results for the existing DFT cabs can be compared to the case where no damping is present in the mounts. These results are as follows in Table 2-8 to Table 2-14.

Table 2-8, Results of the Mathematical Model by Neglecting the Damping Properties of the Current Mount used of the DFT Locomotive cabs.

Mode Number (i)	Frequency, f_i (Hz)
1	5.3717
2	6.7231
3	8.1628
4	18.853
5	20.103
6	28.309

Table 2-9, Mode 1 for Frequency, f_1=5.371(Hz).

Magnitude (Normalised) x10^{-4}	7.6475	90.468	191.6	206.58	1.4462	152.04
Order	(y-dir)	(z-dir)	(x-dir)	(?$_y$-dir)	(?$_z$-dir)	(?$_x$-dir)

Table 2-10, Mode 2 for Frequency, f_2=6.723(Hz).

Magnitude (Normalised) x10^{-4}	155.57	88.652	20.821	19.717	3.4437	152.03
Order	(y-dir)	(z-dir)	(x-dir)	(?$_y$-dir)	(?$_z$-dir)	(?$_x$-dir)

Table 2-11, Mode 3 for Frequency, f_3=8.162(Hz).

Magnitude (Normalised) x10^{-4}	59.2	169.59	42.76	36.857	0.9514	50.474
Order	(y-dir)	(z-dir)	(x-dir)	(?$_y$-dir)	(?$_z$-dir)	(?$_x$-dir)

Table 2-12, Mode 4 for Frequency, f_4=18.853(Hz).

Magnitude (Normalised) x10^{-4}	33.127	0.74088	45.592	44.475	26.741	35.443
Order	(y-dir)	(z-dir)	(x-dir)	(?$_y$-dir)	(?$_z$-dir)	(?$_x$-dir)

Table 2-13, Mode 5 for Frequency, f_5=20.103(Hz).

Magnitude (Normalised) x10^{-4}	38.57	2.9177	35.198	39.236	2.9764	44.539
Order	(y-dir)	(z-dir)	(x-dir)	(?$_y$-dir)	(?$_z$-dir)	(?$_x$-dir)

Table 2-14, Mode 6 for Frequency, f_6=28.309(Hz).

Magnitude (Normalised) $x10^{-4}$	6.6003	2.8801e-3	5.206	6.0273	54.711	7.7747
Order	(y-dir)	(z-dir)	(x-dir)	(?$_y$-dir)	(?$_z$-dir)	(?$_x$-dir)

The stiffness, damping and dimensional parameters used for calculating these values are given in Appendix D.

Chapter 3

3 Experimental Testing and Data Acquisition System

In order to determine the natural frequencies of the cab experimentally it was decided to measure the acceleration at various points in the cab and the under-frame of the DFT locomotives. Before the experimental work could take place, a data capture system needed to be designed to measure and record the accelerations in the DFT locomotive cabs. Initial measurements of the DFT Locomotives, were conducted with a single analogue accelerometer, with a digital output, to determine what system requirements would be necessary for such a data acquisition system. From these initial measurements, of the DFT cabs the power spectral density was obtained. The Root Mean Square (RMS) acceleration, RMS velocity, and RMS displacement were also calculated.

3.1 Initial Measurements

The initial measurements were taken with two, dual-axis accelerometers[1] oriented along 3 orthogonal axes. The accelerometers were then interfaced with the desktop PC, via the XCS40 Spartan Field Programmable Gate Array (FPGA[2]). The FPGA ran on 5V, as this was comparable to the Industry Standard Architecture (ISA) slot connected to the ISA BUS, which also runs on 5V.

The two accelerometers that were used for these initial readings have an analogue output that was then filtered with first order RC filters for each channel. The signal was then fed back into the accelerometer chip where it under-went Duty Cycle

[1] The accelerometer used were ADXL202. Low cost ±2g duel axis, *i*MEMS® accelerometers with digital output, which are manufactured by Analog Devices. The specification sheet has been included in Appendix E.
[2] Some of the specifications for the XCS40 and XCS40-XL have been included in Appendix K.

Modulation (DCM), also know as Pulse Width Modulation (PWM), to produce digital outputs. This approach produces two numbers, one indicated the time when the signal was high (T_1) and the other indicated the time when the signal was low (T_2-T_1). The sum of these two values gave the total period of the signal T2. The acceleration is proportional to the length of time that the signal was high (T_1). The corresponding formula from the accelerometer specification sheet in Appendix E is shown as, Acceleration(g)=(T_1/T_2-0.5)/12.5%.

The signal from the accelerometers was sent to the FPGA at a frequency of 5kHz. The FPGA, which has an independent clock, then receives a signal from the PC at a frequency of 300Hz to sample the signal from all of the accelerometer channels. These samples are then sent directly back to the PC. The signals are then saved directly to a binary file for each of the three channels that are used for x, y, and z. The time that the signal was high, T1, and the time that the signal was low, T2-T1, is recorded.

From the initial measurements several quantities were calculated. Specifically, the Power Spectral Density (PSD), RMS values, and real time measurements for the accelerations, velocities, and displacements. The jerk[3], which is the rate of change of acceleration due to impact loading, was also calculated.

PSD plots of the locomotive data from the tests that were conducted are shown in Appendix F. From these plots it appears that the 150Hz signal, which was the maximum possible range of measurement due to the way the system was configured to allow a maximum sampling frequency of 300Hz, did not give enough information about the higher frequency range.

At the same time measurements were also made with an analogue Brüel & Kjær, type 4344 accelerometer with a measuring range of ±50g's. This accelerometer was interfaced with a YOKOGAWA 4-channel digital oscilloscope via a Brüel & Kjær dual charge amplifier and integrator. Measurements were taken simultaneously with the original two accelerometers. With the two different set-ups, a comparison could

[3] This is the rate of change of acceleration, with respect to time.

be made between the two dual axis accelerometers to see if they were sufficient in recording the maximum accelerations as well as the maximum upper frequency that is of significance to the vibration that affects humans. These tests showed that the cheaper dual axis accelerometers were sufficient for the testing that was needed. Therefore when the full data acquisition system was designed, it was decided that the minimum sampling frequency would be 1000Hz. This value would allow for access to vibrational information, up to 500Hz due to the Nyquist sampling frequency principle.

From the literature that was reviewed on WBV exposure of humans in the work place [Wilde, 1980], it was unnecessary to measure the vibration that occurs above 100Hz, as frequencies above 100Hz were not shown to have significant effects on the human body.

3.2 Design of a Full Data Acquisition System

In order to compare the natural frequencies of the theoretical 'rigid body' mathematical model consisting of six degrees of freedom to the natural frequencies that are measured on the DFT locomotive cab, six measurements would need to be made. This data could then undergo a coordinate transformation so that the vibration in the principal coordinates, consisting of x, y, z, θ_x, θ_y, and θ_z from the mathematical model, could be compared to the experimental data. For this task to be accomplished a data capture system capable of simultaneously collecting a minimum of six signals at 1 kHz is required. To calculate the energy that was being passed through from the under-frame of the cab meant that another six measurements were required from the under-frame. This data also allowed auto-correlations and cross-correlations to be performed.

Tri-axial measurements of the accelerations experienced at the driver's seat were also considered necessary. This data could be used to judge the vibrations that were being directly transmitted to the driver of the locomotive. This requirement meant a further 3 channels were needed, raising the number of channels to fifteen. To allow for

unexpected measurements that may be required and to allow for additional readings that would be required in order to verify the 'rigid-body' assumption the data acquisition system was designed for a maximum of twenty digital channels. It was decided that an FPGA was one of the few viable solutions for collecting this data with each of the twenty channels sending digital data simultaneously at 5kHz, as it is one of the few hardware products capable of collecting digital data in such large quantities at that rate for the price that was paid.

A 486 laptop with a USB port running at 3.3V was used for the full data acquisition system. The XCS40XL Spartan-XL FPGA was used in conjunction with this laptop as it was compatible with the 3.3V that was required. The speed that could be obtained from the FPGA and the cost involved in purchasing hardware made this configuration a sensible option.

3.3 Accelerometer Sensors

The accelerometers chosen are the ADXL202 low cost ±2g, dual axis, *i*MEMS® accelerometers with digital outputs. One of the main considerations in the selection of accelerometers was the cost due to the large number of sensors that were necessary for the experimental procedure. As well as being an inexpensive option the fact that the accelerometers produce a digital output is important due to the presence of significant electrical noise in the locomotive environment.

The ADXL202 accelerometers do have a base noise level. However, the accelerometer has a Duty Cycle Modulator (DCM) or Pulse Width Modulation (PWM) on board of the chip, which eliminates the possibility of further noise being introduced to the signal as it travels down the wire to the FPGA. The resolution of the PWM is approximately 14-bit accuracy although the FPGA has two 16-bit registers for the acceleration data. The actual resolution obtained from the accelerometers after noise is accounted for is close to 8-bits due to the 13.7mg RMS noise present. Full details on the specification of the accelerometers can be seen in Appendix E.

The printed circuit-board diagrams produced for the accelerometers can be seen in Appendix G and the schematic included in Appendix H. The accelerometers were super-glued firmly to the circuit-boards, to eliminate any chance of the accelerometer chips and the pins of the chip from acting as a spring mass system. The circuit boards were then mounted to a steel block and encapsulated in electrical quality epoxy resin to protect them from the harsh locomotive environment. A picture of the accelerometer set-up can be seen in Figure 3-1.

Figure 3-1, Modular creation of two of the accelerometer blocks, mounted above a DFT cab mount.

As Figure 3-1 shows, the accelerometer blocks were created in a modular form, so that any combination of accelerometer block could be connected and 5 sets of 3 axes orthogonal units could be used on the locomotive.

Testing was performed on the ADXL202 accelerometers to see how linear the acceleration measurements were outside its working range of ±2g. This test was performed on a shaker table that controls the frequency and displacement input in the form of a sine-wave. An analogue Brüel & Kjær, type 4344 accelerometer, with a measuring range of ±50g's was used as a control, in order to establish the accuracy of the results of the ADXL202 accelerometers.

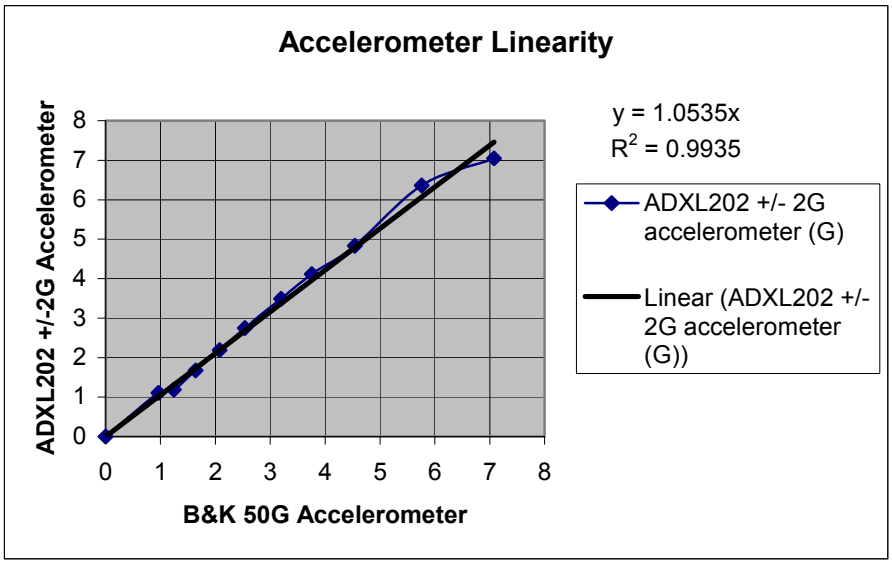


Figure 3-2, Linearity of ADXL202 Accelerometer.

Although the accelerometer recommended working range was ±2g it was considered acceptable from the results of this experiment to use the accelerometers to ±7g as seen in Figure 3-2. Results were unable to be obtained above ±7g as the ADXL202 would not produce meaningful results. The experiments were performed by maintaining a constant frequency of 15 Hz. The exact corresponding frequencies for the measured accelerations are give in Table 3-1.

Table 3-1, Results of ACXL202 linearity test.

B&K 50 G accelerometer (G)	B&K 50 G accelerometer, Frequency Measured (Hz)	ADXL202 +/- 2G accelerometer (G)	ADXL202 +/- 2G accelerometer, Frequency Measured (Hz)
0	15	0	15
0.96	16.23	1.111	14.94
1.24	15	1.185	14.947
1.64	15.1	1.67	15.1
2.08	15.15	2.1865	15
2.54	15	2.75	15.13
3.2	15	3.49	14.99
3.76	15.2	4.1164	14.9
4.54	15.17	4.8331	15.15
5.76	15.04	6.359	14.97
7.08	15	7.0492	15.17

Although accelerations experienced on the DFT Locomotive were not expected to exceed ±2g, was reassuring to know that there was a buffer present for recording higher accelerations should they occur.

3.4 Duty Cycle Modulation (DCM) for A/D Conversion

The frequency of the DCM is 5kHz which is the upper limit of the DCM. The DCM works on the principle of Pulse Duration Modulation (PDM). This approach is dependent on the duration of time that the output remains high. The digital output is oscillating and is on for a percentage of the time proportional to the amplitude of the analogue signal being measured and is off for the remainder of the time. When this signal is integrated over time the effective output is proportional to the percentage that the output was held high.

3.5 Field Programmable Gate Array (FPGA)

The FPGA is the hub of the data acquisition system. The full specifications for the FPGA cannot be included, as they are very large, but a small section has been included in Appendix I. The FPGA printed circuit-board diagram is included in Appendix J. It should be noted from the circuit diagram that the main parts of the board consist of the FPGA chip and the Micro-controller chip. The Micro-controller has the USB port and is used to interface between the FPGA and the laptop, as the FPGA does not talk to laptop directly.

3.6 FPGA Programmable Hardware

The programmable hardware of the FPGA purely configures the logic gates necessary to perform the operations of sampling the digital signals. There are twenty channels providing digital signals from the accelerometers that are sampled on the rising edge of the FPGA clock pulse. This information is then sent to the laptop via the Micro-controller on the FPGA board. The schematics for the programmable hardware can be seen in Appendix K.

3.7 Software

The software for the Micro-controller[4] handled the protocol for the FPGA and its initial set-up conditions. The software then downloads the software to run on the FPGA. The last section of software then handles the processing of the information sent from the FPGA and sends the information up the USB port to the laptop. The software for the PC[5] performed a range of activities. However the main purpose of the software was to receive data from the USB port, check that all the information was

[4] The name of this software program is Target.c
[5] The name of this software program is Main.c

received and save the information as hexadecimal binary files. The listing of these programs can be seen in Appendix L.

MATLAB was used for the post-processing and analysis of the experimental data that was collected from the FPGA. MATLAB is a commercial numerical analysis package. The MATLAB software suite of programs used for the data acquisition are provided in Appendix M.

Directory of files:

- *full_data_aquthesis.m*: This is the main calling program of the post processing and analysis of the experimental results. It calls all of the functions that follow.

- *full_input_filesthesis.m*: This function loads all of the files that were created from the experimental trials and stores them in a cell array, so that they can be calibrated later.

- *hex_accelthesis.m*: This function is called by full_input_filesthesis.m to transform the hexadecimal binary data files, which were created from the FPGA and data acquisition system into the uncalibrated acceleration data that full_input_filesthesis.m can read.

- *fill_FPGA_gapsthesis.m*: This function is called up by hex_accelthesis.m to interpolate data that was missed by the laptop when the information was sent by the FPGA.

- *clear_shock_accelthesis.m*: This function is called up by hex_accelthesis.m to eliminate any shock loads outside the measurement scale that the accelerometers have experienced. These shock loads are recorded as infinite. They are replaced by interpolating the surrounding data points to replace these stray measurements.

- *clip_accelerationthesis.m*: This function is called up by hex_accelthesis.m to eliminate any measurements outside the possible accuracy of the accelerometers, and then replace it with the maximum possible value.

- *is_smoothesis.m*: This function is called up by hex_accelthesis.m and is used to apply a zero-phase forward and reverse butterworth 3rd order low pass filter with a cutoff at 500Hz.

- *iso2631_weighted_filter_Wa.m*: This function is called up by hex_accelthesis.m as a specially weighted band pass filter.

- *full_cal_files2thesis.m*: This function loads all of the calibration files that are needed for the purposes of calibrating the experimentally collected data.

- *full_G_calibrationthesis.m*: This function is used to calibrate the data, and ensure that the axes of the accelerometers are orthogonal to each other by applying a rotation matrix.

- *full_velocitythesis.m*: This function is used to integrate the acceleration.

- *full_displacementthesis.m*: This function is used to integrate the velocity.

- *full_quintons_PDSthesis.m*: This function calculates the Power Spectral Density (PSD).

- *full_jerkthesis.m*: This function calculates the jerk (rate of change of acceleration).

- *full_RMS_normthesis.m*: This function calculates the RMS acceleration, RMS velocity and the RMS displacement.

3.8 Hardware Filters and Software Filters

The hardware filter used was a first order RC filter. The resistance R_{filt} (32kΩ) of the filter was internal to the accelerometer chip and therefore could not be changed. Unfortunately, this predefined the resistance in the accelerometer chip for the purpose of filtering had an inherent error of 25%. The capacitors (0.01μF) that were used for the purpose of filtering also had an inherent error as large as 20%. These errors meant that the cut-off frequency of the filter was not reliable and could vary from 345Hz to 777Hz.

For this reason it was considered necessary to include a software filter that was far more accurate. The filter weighting design used was a standard and accepted method of filtering for such data in these types of applications. The filters that were used are shown in Figure 3-3 to Figure 3-6, where both the theoretical (green line) and the implemented (blue line) filter curves are shown.

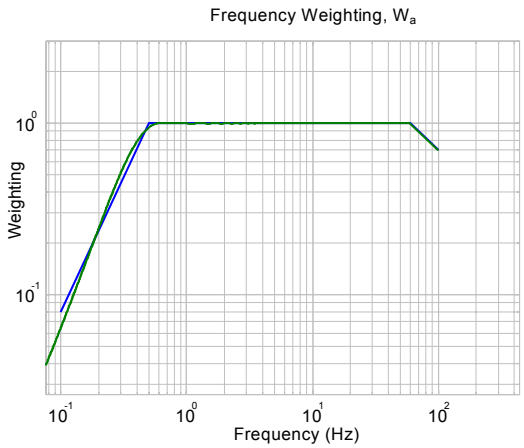


Figure 3-3, Standard band-pass filter, used for ride and passenger comfort in seated position and passenger comfort in standing position.

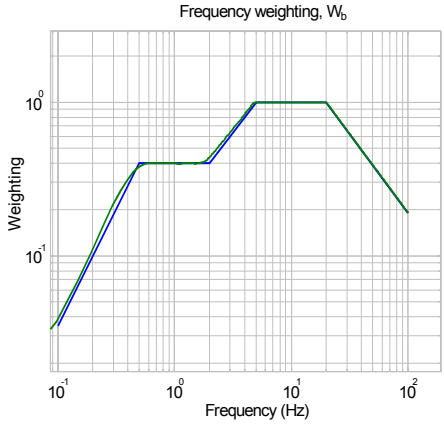


Figure 3-4, Filter used for vertical measurements made from the floor of the cab for the purpose of ride and passenger comfort in seated and standing positions.

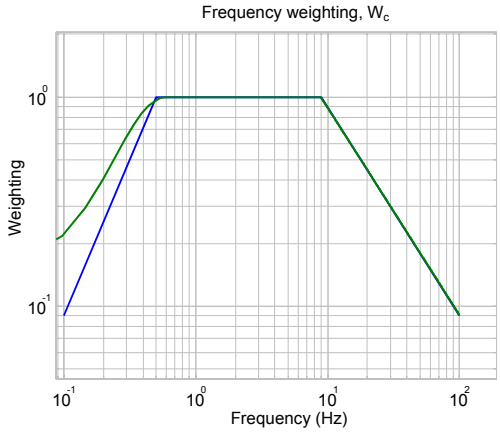


Figure 3-5, Filter used for acceleration of longitudinal direction for passenger comfort in the seated direction.

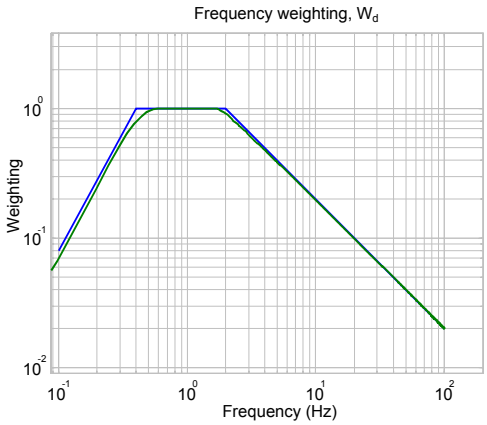


Figure 3-6, Filter used for ride comfort for the longitudinal and lateral directions, passenger comfort for the seated position in the lateral direction and passenger comfort in the standing position for the lateral and longitudinal directions.

These filters were used as zero-phase forward and reverse digital filters. Hence, after filtering in the forward direction the sequence is reversed and run back through the filter. The final output is the time reverse of the initial output of the second filtering operation. The result has precisely zero phase distortion and a magnitude modified by the square of the filter's magnitude response. Care is taken to minimize start-up and ending transients by matching initial conditions.

The filters shown in Figure 3-3 to Figure 3-6, are derived from 'ISO 2631: Evaluation of human exposure to whole body vibration', specifically for the application railway vehicles.

3.9 Verification Test (Cantilever Beam)

It is important to determine the accuracy of the measurements using the data acquisition system and measurement apparatus. For this verification it was decided to use the entire system to measure the natural frequencies of a cantilever beam and compare the results obtained with the calculated frequencies. The frequencies were measured by conducting a series of impulse tests on the cantilever.

The accelerometer was attached to the end of the beam. The associated mass of the accelerometer block was taken into account when the theoretical natural frequencies were calculated. The first two experimental and theoretical natural frequencies were the same to within one decimal place for the first natural frequency, and within 2.0 Hz for the second natural frequency. This result is seen in the PSD graph of Figure 3-7, and the zoom of the first natural frequency in Figure 3-8. Although far more than the first two natural frequencies could be calculated theoretically, experimentally higher natural frequencies were unobtainable due to noise that was present. The Power Spectral Density PSD was unable to register to any accuracy where the peaks for the fifth and higher natural frequencies occurred.

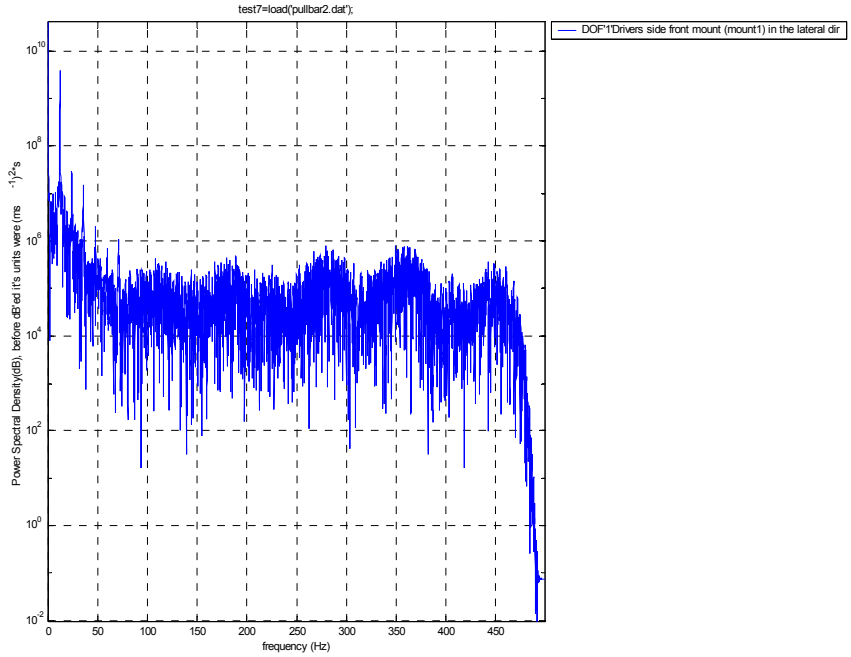


Figure 3-7, Denotes the accuracy of the experimental natural frequencies.

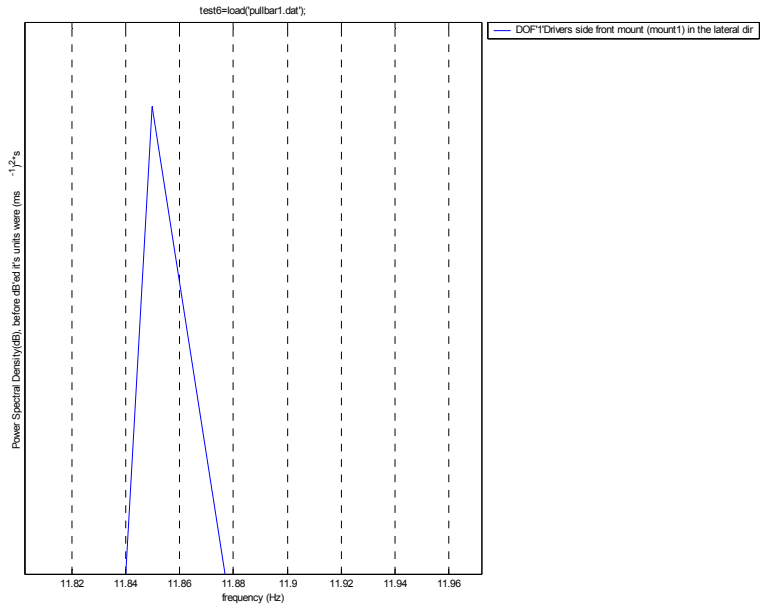


Figure 3-8, Zooming in on the first natural frequency.

The theoretical natural frequencies were calculated with a Rayleigh Ritz method, program 'RR26_6_00.m', in Appendix D. The theoretical natural frequencies calculated were as follows:

f_1=11.87Hz
f_2=32.407Hz
f_3=118.75Hz
f_4=288.98Hz

As seen from Figure 3-8, the experimental result is only 0.03Hz different from the experimental result. The discrepancy between the calculated and measured value of the second natural frequency was greater than the discrepancy of the first mode. This increase in noise and experimental error with the increase in the frequency measured is due to the low-cost accelerometers that were used.

As seen from Figure 3-7, it was difficult to produce a graph of the natural frequencies orders of magnitude above the noise for the system due to the fact that is quite difficult to produce a 'good' impact test, as the beam can not be struck too hard or too much noise is obtained that 'swamps' the system. If it is not struck hard enough it is difficult to excite the higher frequency modes of the system.

Chapter 4

4 Data Analysis

The purpose of this Chapter is to discuss the data analysis methods and comparison of the theoretical and experimental results. The theory and the simulation were discussed in Chapter 2. In Chapter 3 the design of the experimental tools required for this work was discussed. However, neither of these two chapters covered how the knowledge gained from the theory, simulation and experiments were implemented or how the results are to be compared.

4.1 Experimental Set-Up

For reference, the figure of Chapter 2 has been repeated as Figure 4-1. The 6 major coordinates that were required in order to make the coordinate transformation of x, y, z, θ_x, θ_y, and θ_z have been shown below with bold arrows and the labelling of y_2, z_2, x_3, y_3, z_3, z_4, x_i, y_i, and z_i. For example, the letter 'x' indicates the direction of the coordinate that was measured from the mounted accelerometer. The subscript number or letter that follows the vectorial direction indicates the position of the measurement from the mounted accelerometer.

To verify and implement the rigid body assumptions the acceleration along coordinates x_1, y_1, z_1, x_2 , x_4, and y_4 were also collected. The typical experimental set up is shown in Figure 4-2 though to Figure 4-4. Figure 4-2 shows the housing of the FPGA, and in the top right hand corner of the photograph is the base of the drivers console. The attachment and positions of the sensor both directly under the drivers' seat and on rear driver side mount is shown.

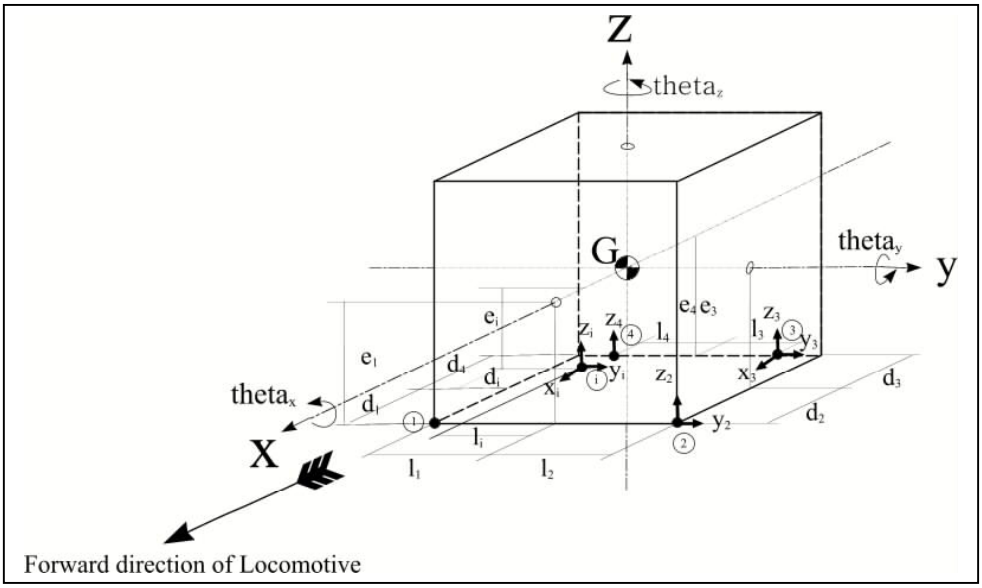


Figure 4-1, Schematic of the DFT Locomotive cab.

Figure 4-2, Experimental set-up from the rear of the cab on the driver's side.

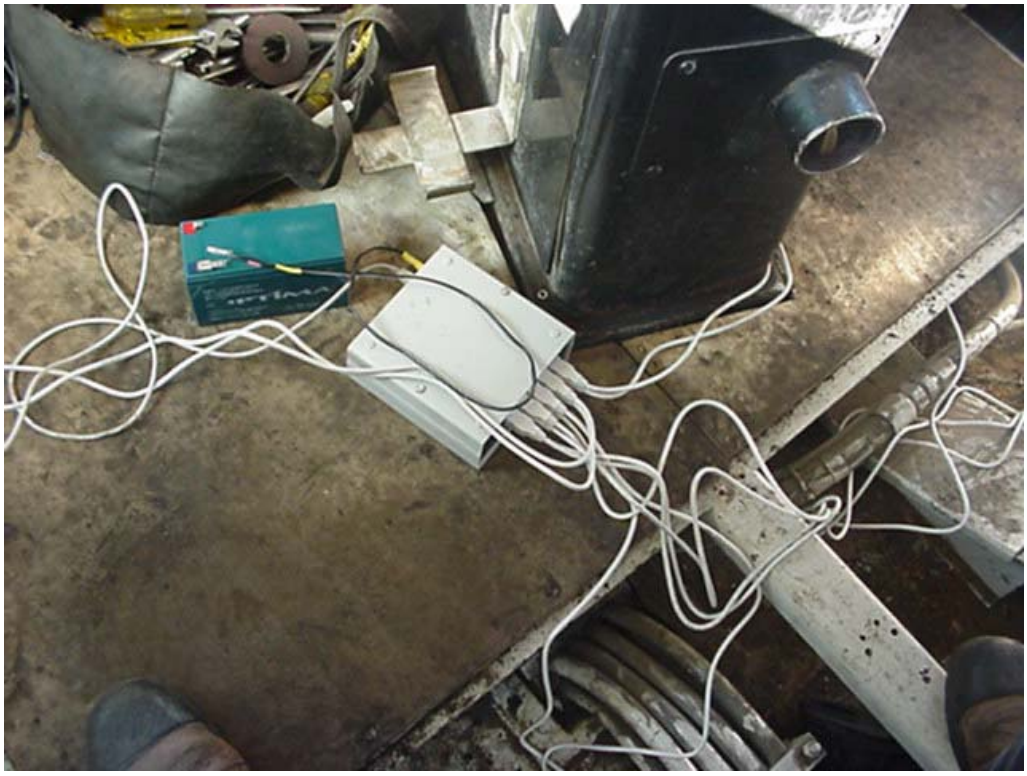


Figure 4-3, Experimental set-up from the rear of the cab on the passenger side.

Figure 4-4, Experimental set-up under the drivers seat.

4.2 Apparatus Attachment to Locomotives

It was not possible to rigidly mount the accelerometer block to the locomotive to take the acceleration measurements required from the locomotive cabs and under-frames. A mechanical fastener such as a bolt was not practical as it was unacceptable to drill half-a-dozen holes in half of Tranz Rail's DFT fleet. The most practical way of mounting the accelerometers to the locomotive cab, and the different frequency responses obtainable from different methods of mounting the accelerometers was determined. The most acceptable way to mount the accelerometers was using strong magnets. Rare earth magnets with a composition of Neodymium-Iron-Boron were used.

The retaining force of the magnets used was experimentally determined with the use of an electro-magnetic shaker. An accelerometer block, as seen in Figure 3-1 was placed on the shaker table. The tests were performed by setting the amplitude and frequency of the shaker table to its minimum settings. The accelerometer would then record data for a 20 second sample. Once the sample was complete the frequency was incremented and the test was performed again until the upper frequency range of the accelerometer was reached at 1000Hz. The amplitude of the shaker table was then incremented and the tests were performed again throughout the entire frequency range. As the amplitude of the shaker table was incrementally increased, a distinctive chatter could be heard and it soon became apparent that the magnet was losing contact with the surface of the shaker table. The acceleration at which the accelerometer was no longer fixed to the shaker table was outside of the measurable ±7g range of the accelerometer. Hence, the actual acceleration at which the accelerometer released is unmeasurable due to the fact that the amplitude dial on the shaker table was unable to indicate the acceleration experienced. However, the frequency of 550Hz, was the indicated frequency of the shaker table. This frequency is also outside of the workable range for the accelerometer of 475Hz.

4.3 Confirmation of the Rigid Body Assumption of the Mathematical Model

To check the flexural rigidity of the locomotive cab to verify the rigid body assumption, the acceleration along coordinates x_1, y_1, z_1, x_2 , x_4, and y_4 were also recorded. The relationship between the coordinates y_2, z_2, x_3, y_3, z_3, z_4, x_i, y_i, and z_i with x, y, z, θ_x, θ_y, and θ_z at the centre of mass is expressed by the matrix relationship in Equation (4-1) through Equation (4-3) where the last equation solves for the new coordinates.

$$\underset{b}{\begin{bmatrix} z_3 \\ y_3 \\ x_3 \\ z_2 \\ y_2 \\ z_4 \end{bmatrix}} = \underset{A}{\begin{bmatrix} 0 & 0 & 1 & l_3 & d_3 & 0 \\ 0 & 1 & 0 & e_3 & 0 & -d_3 \\ 1 & 0 & 0 & 0 & -d_3 & l_3 \\ 0 & 0 & 1 & l_2 & -d_2 & 0 \\ 0 & 1 & 0 & e_2 & 0 & d_2 \\ 1 & 0 & 0 & -l_4 & d_4 & 0 \end{bmatrix}} \underset{x}{\begin{bmatrix} x_G \\ y_G \\ z_G \\ \theta_{x_G} \\ \theta_{y_G} \\ \theta_{z_G} \end{bmatrix}}$$

Equation (4-1)

$$\underline{b} = A\,\underline{x}$$

Equation (4-2)

$$\therefore\ \underline{x} = \{A^{-1}\}\,\underline{b}$$

Equation (4-3)

It then seemed reasonable to be able to make a coordinate transformation from the principal coordinates of the mathematical model to a completely different set of measured coordinates. This transformation enables the prediction of the acceleration at any point of the cab of the locomotive.

The relationship between the coordinates x_1, y_1, z_1, x_2 , x_4, and y_4 with x, y, z, θ_x, θ_y, and θ_z is expressed in Equation (4-4) and Equation (4-5).

$$\underset{c}{\begin{bmatrix} z_1 \\ y_1 \\ x_1 \\ x_2 \\ y_4 \\ x_4 \end{bmatrix}} = \underset{B}{\begin{bmatrix} 0 & 0 & 1 & -l_1 & -d_1 & 0 \\ 0 & 1 & 0 & e_1 & 0 & -d_1 \\ 1 & 0 & 0 & 0 & -e_1 & d_1 \\ 1 & 0 & 0 & 0 & -e_2 & -l_2 \\ 0 & 1 & 0 & e_4 & 0 & -d_4 \\ 1 & 0 & 0 & 0 & -e_4 & l_4 \end{bmatrix}} \underset{x}{\begin{bmatrix} x_G \\ y_G \\ z_G \\ \theta_{x_G} \\ \theta_{y_G} \\ \theta_{z_G} \end{bmatrix}}$$

Equation (4-4)

Hence:

$\underline{c} = B\,\underline{x}$

Equation (4-5)

Therefore, the measured acceleration of the vector $\underline{b}$ can be used to determine the desired acceleration vector, $\hat{\underline{c}}$, with Equation (4-6).

$\hat{\underline{c}} = B\{A^{-1}\}\underline{b}$

Equation (4-6)

Where the estimate $\hat{\underline{c}}$ is within a small percentage error of $\underline{c}$. However, the predicted accelerations had larger than desired errors as demonstrated in the Figure 4-5 though Figure 4-7.

Due to the percentage error shown in these figures the transformation matrices and software program used to determine other measured coordinates were tested with some simple test data. This data assumed that the body was perfectly rigid. Sinusoidal data was used to test the x, y, z, θ_x, θ_y, and θ_z coordinates individually. This test data was then run through the program used for the estimation of response. Results showed that the percentage error of every coordinate was within the machine accuracy of the computer.

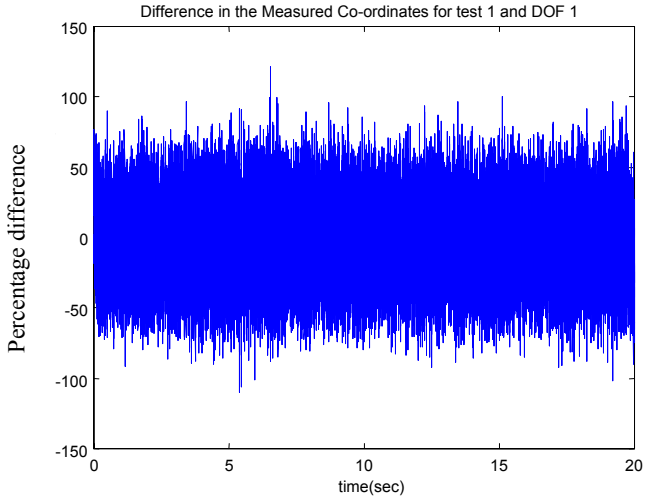


Figure 4-5, Percentage Error of the estimation of the Drivers side front mount (mount1) in the lateral direction.

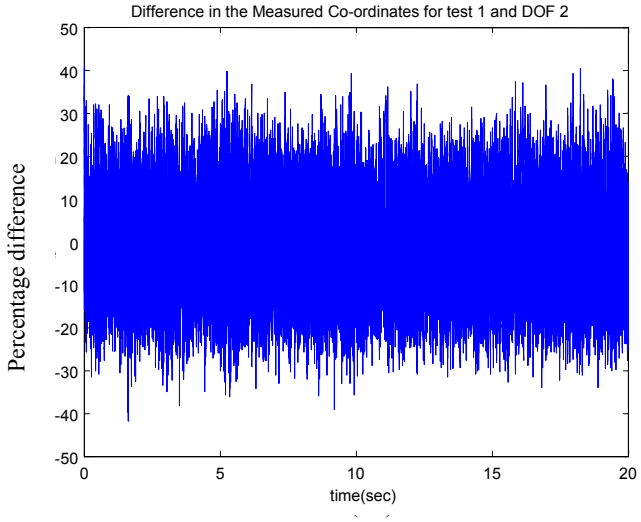


Figure 4-6, Percentage Error of the estimation of Drivers side front mount (mount1) in the longitudinal direction.

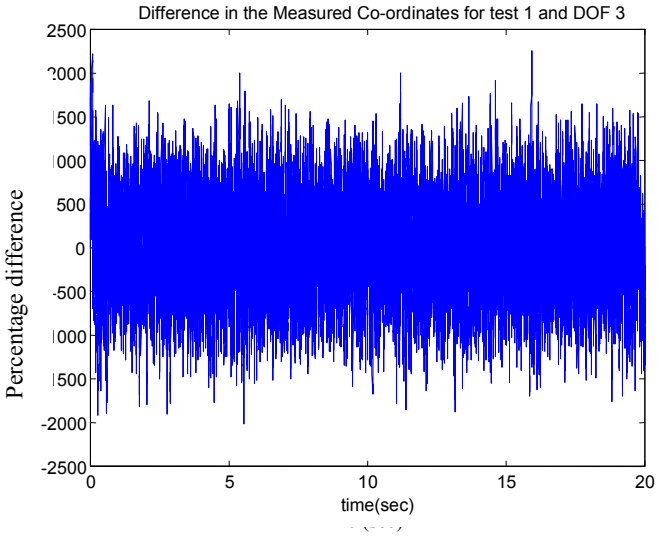


Figure 4-7, Percentage Error of the estimation of Drivers side front mount (mount1) in the vertical direction.

From this result it is concluded that the motion of the cab of the locomotive is not fully rigid.

Different scales were used for Figure 4-5 to Figure 4-7 because the range of percentage error varies with the direction of measurement, hence making each plot clearer. The greater percentage error seen in Figure 4-7 may be due to the greater flexibility of the floor grillage in the out-of-plane direction.

4.4 Measured effectiveness of DFT Locomotive mounts

It was noted that the results of the natural frequencies of the damped and undamped system were very similar due to the low damping value that was used. A damping ratio, ζ, of 0.0385 was calculated. The static, axial and radial stiffness were 1575kN/m and 5512kN/m respectively. The dynamic axial and radial stiffness were 1968kN/m and 6891kN/m respectively. The dynamic values were used for the purpose of the mathematical model.

Due to the uncertainty of the cab mount stiffness values, tests were performed to measure their transmissibility. The experimental set-up was identical to the set-up outlined in Section 4.1. Accelerometers were placed on both sides of the mounts to measure the following coordinates, y_2, z_2, x_3, y_3, z_3, z_4.

From the measured accelerations auto-correlations are produced for the accelerations taken from all of measurements of the underframe side of the mounts. The cross-correlations with the underframe side of the mounts as the inputs and the cab side of the mounts as the outputs are also compared. It is observed for each measurement that the auto-correlation and the cross-correlations were almost identical when overlaid.

This result indicates that the current mounts used in the cabs are not actually isolating the cab from the locomotive and that the mounts are acting as rigid connections.

As seen in Figure 4-8 through Figure 4-13 the cross-correlation is in fact greater than the auto-correlation, indicating that the mounts are actually magnifying the vibrational acceleration.

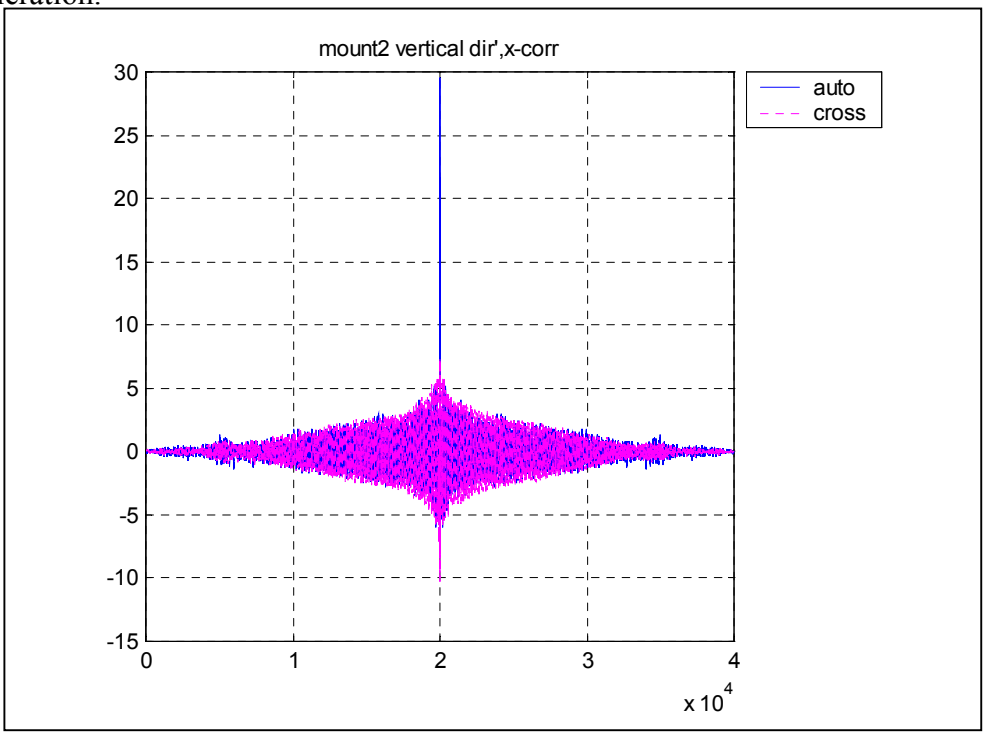


Figure 4-8, Passenger side front mount, Vertical direction.

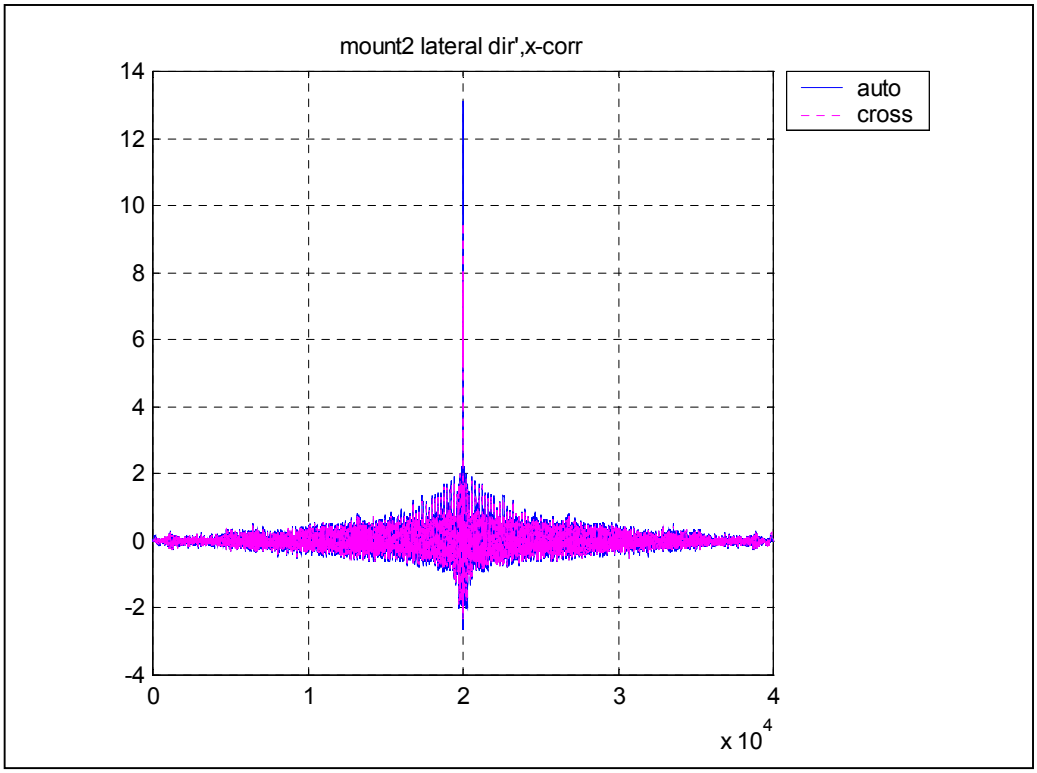


Figure 4-9, Passenger side front mount, Lateral direction.

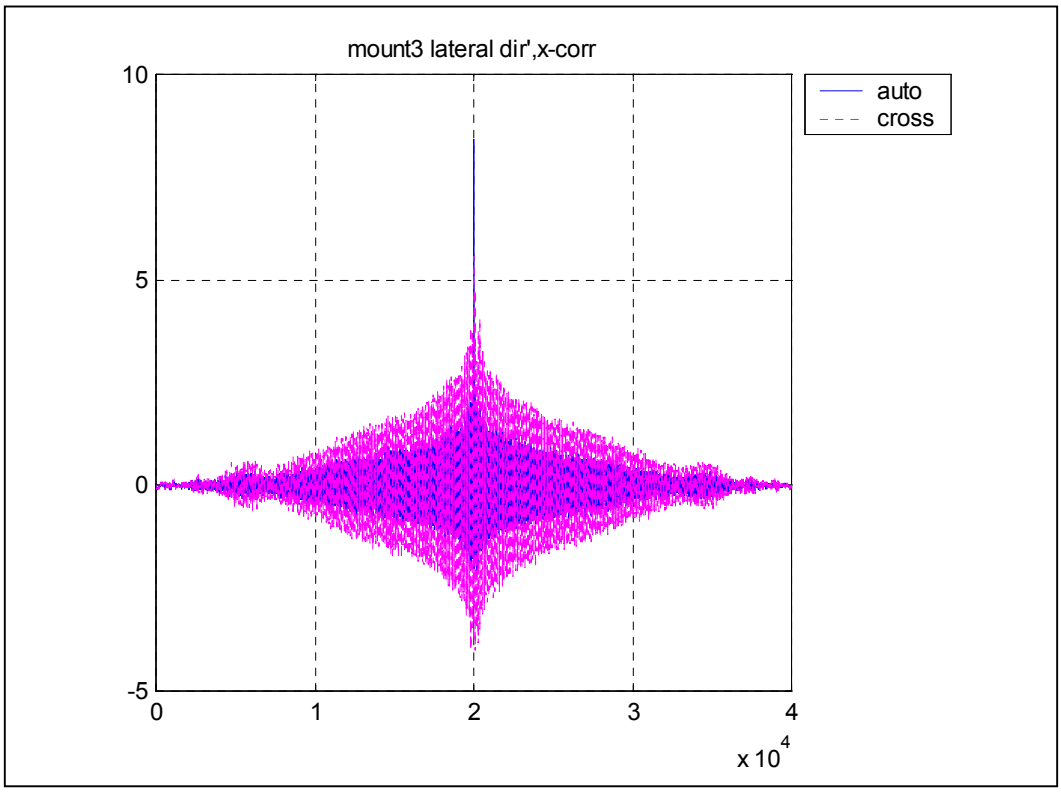


Figure 4-10, Passenger side rear mount, Lateral direction.

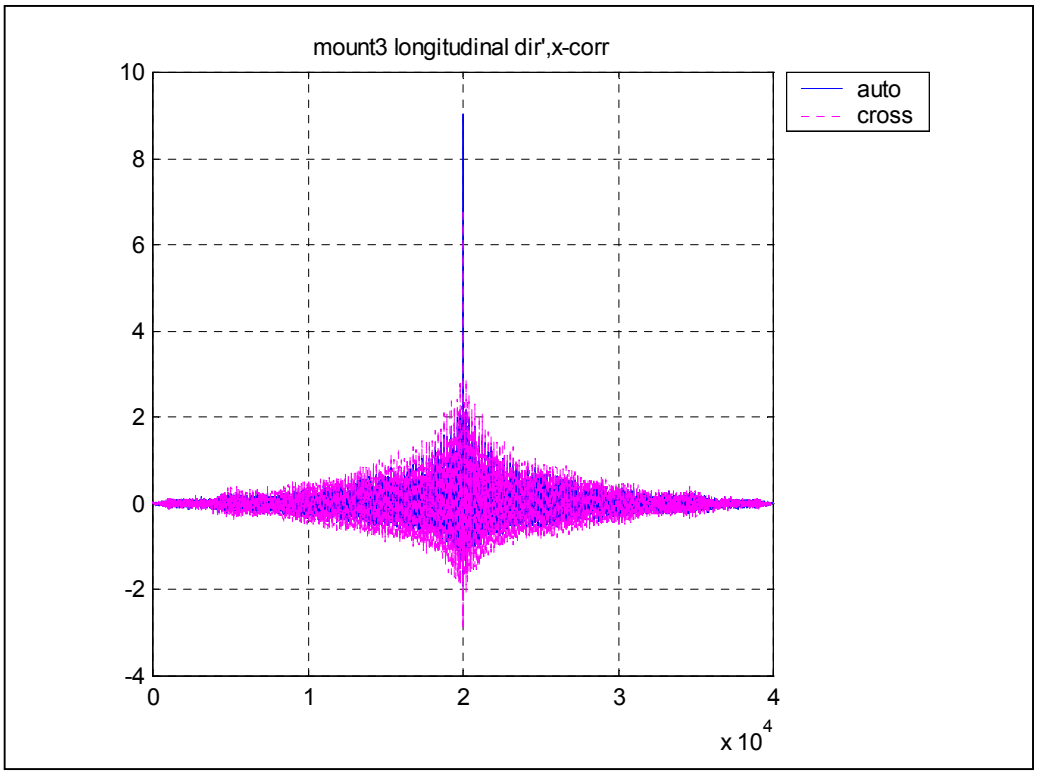


Figure 4-11, Passenger side rear mount, Longitudinal direction.

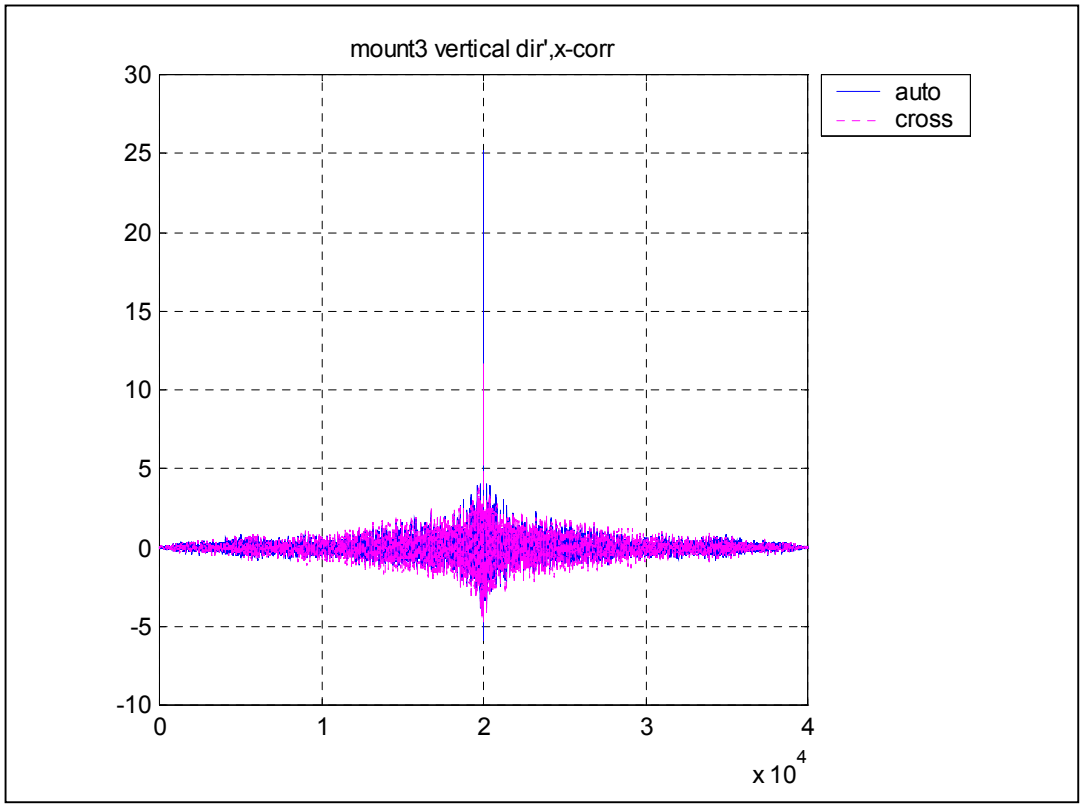


Figure 4-12, Passenger side rear mount, Vertical direction.

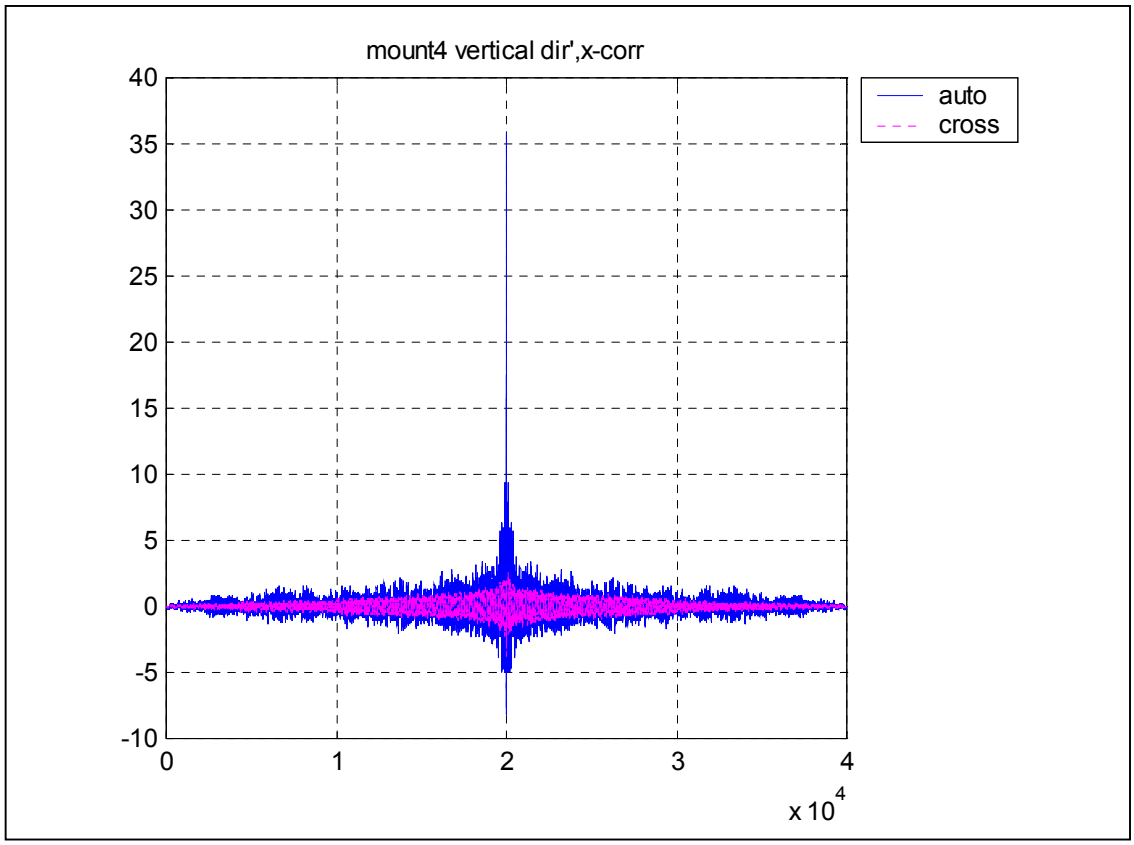


Figure 4-13, Drivers side rear mount, Vertical direction.

These results can be compared with the correlation between the acceleration measured at the seat and at each of the four mounts. Figure 4-14 shows the correlations between the vertical acceleration of mount 4 and the vertical acceleration of the seat. The close to perfect match between the acceleration at the seat and the mounts indicates approximately 100% transmission of vibration through the cab to the seat.

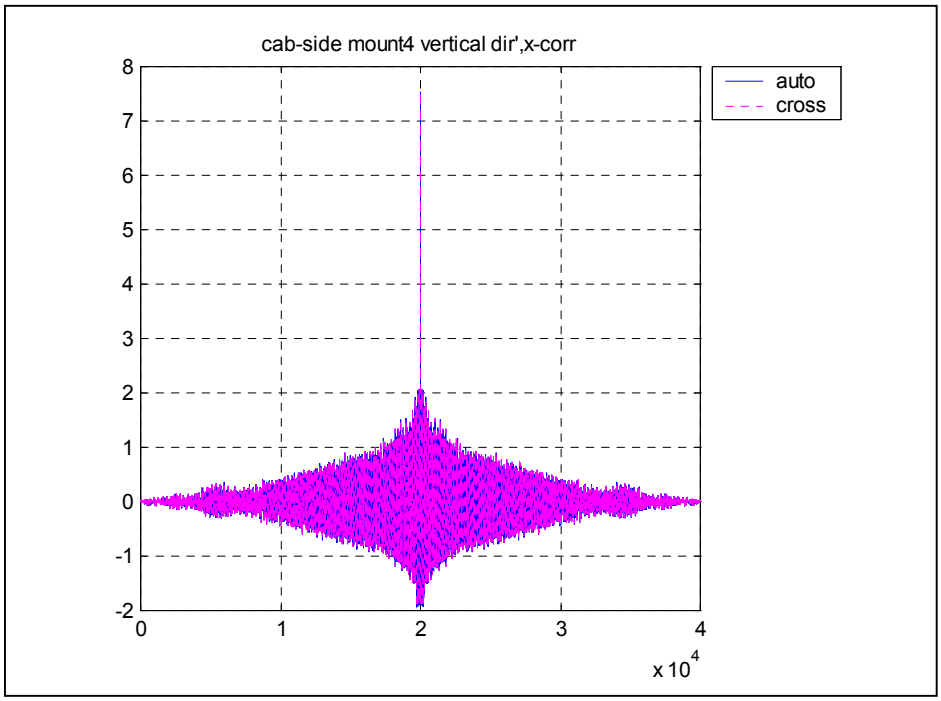


Figure 4-14, Relationship between vertical acceleration of the seat and the mounts.

4.5 Experimentally Determining the Natural Frequencies of the Locomotive Cab.

To confirm the theoretical natural frequencies of the DFT locomotive cab it was decided that if the cab could have an impact load imposed on it with the acceleration measured in the x, y, and z coordinates at all four mount positions, a PSD of each measurement would reveal high peaks indicating the possibility of natural frequencies. Initially, the locomotive cab was excited while stationary by having

another locomotive collide into it with sufficient force. However, this approach did not work well, as it was difficult to produce sufficient force to cause the cab to rock independently from the rest of the locomotive. This result was confirmed when the results were post processed and the PSD with respect to frequency showed very little distinguishing information. It was hypothesised that this result is due to the fact that the system was excited through its supports. For this reason more tests were conducted. However, this time the cab body was excited directly with the cab being subjected to a 115kg weight being dropped from sufficient height on the cab of the locomotive.

This revised test procedure produced sufficient vibration of the cab and many significant peaks were visible. An illustration of the results obtained from the PSD of the x, y, and z direction for the drivers-side front mount and the passengers-side mount is shown Figure 4-15 though to Figure 4-20. A summary of the frequency peaks obtained is given in Table 4-1. All following power spectral density plots are scaled to fit, though the 'Power Summaries' of these plots in Chapter 5 are consistently scaled for a more direct comparison.

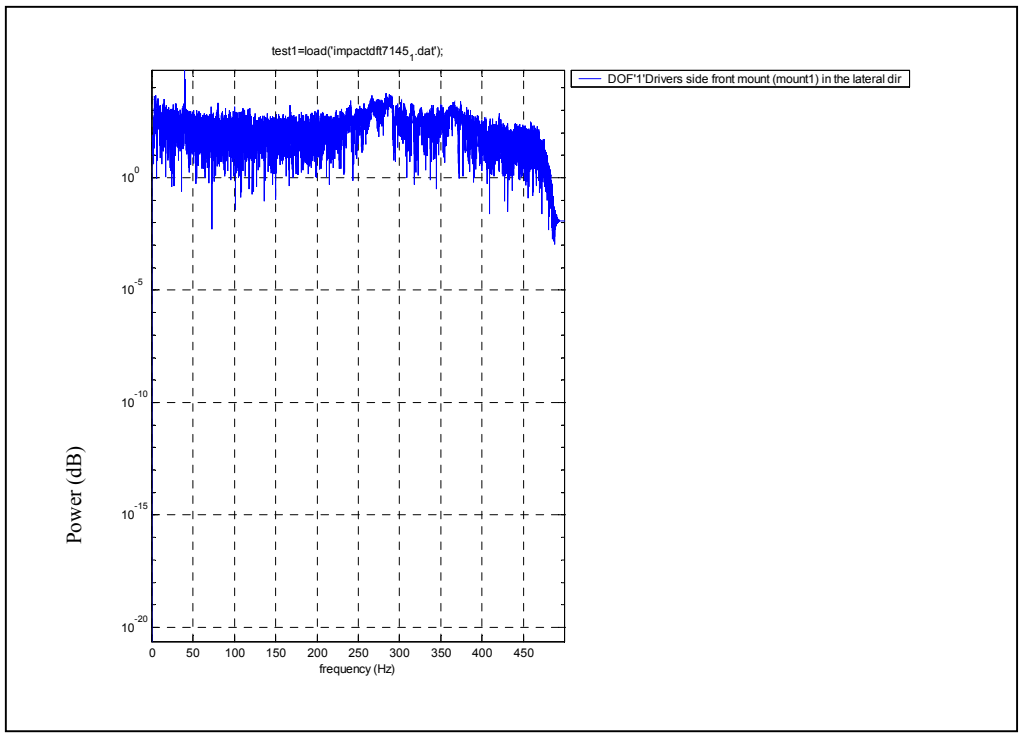


Figure 4-15, DOF1, Drivers side, Front Mount (Mount 1), Lateral Direction.

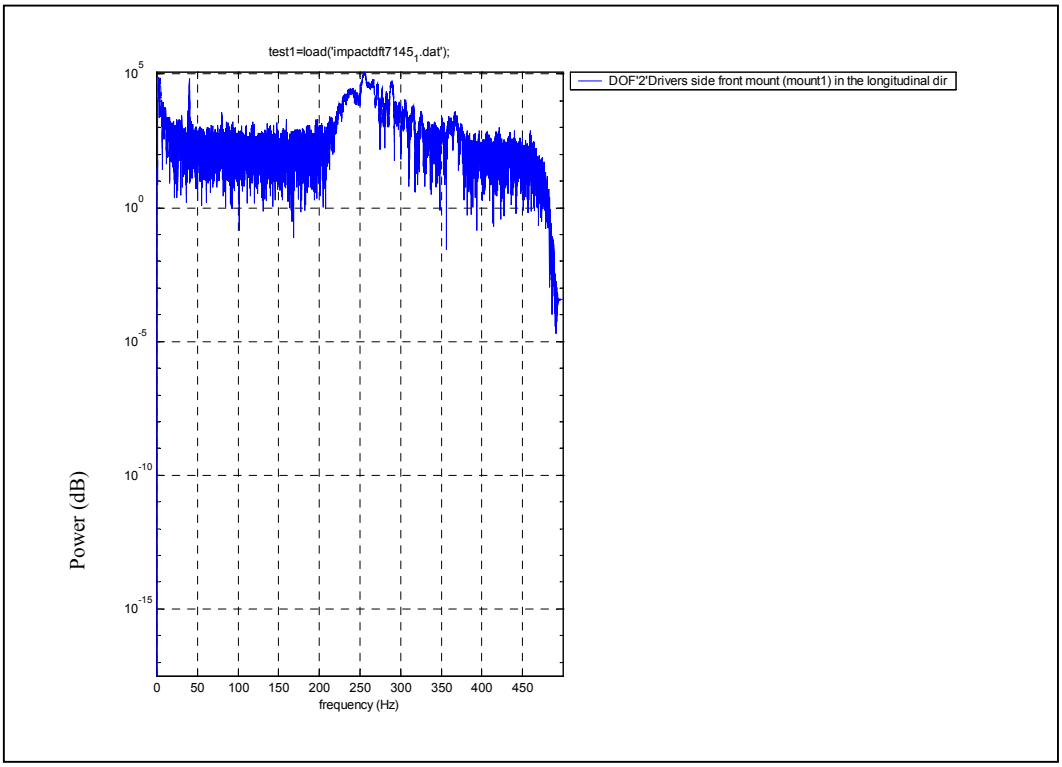


Figure 4-16, DOF2, Drivers side, Front Mount (Mount 1), Longitudinal Direction.

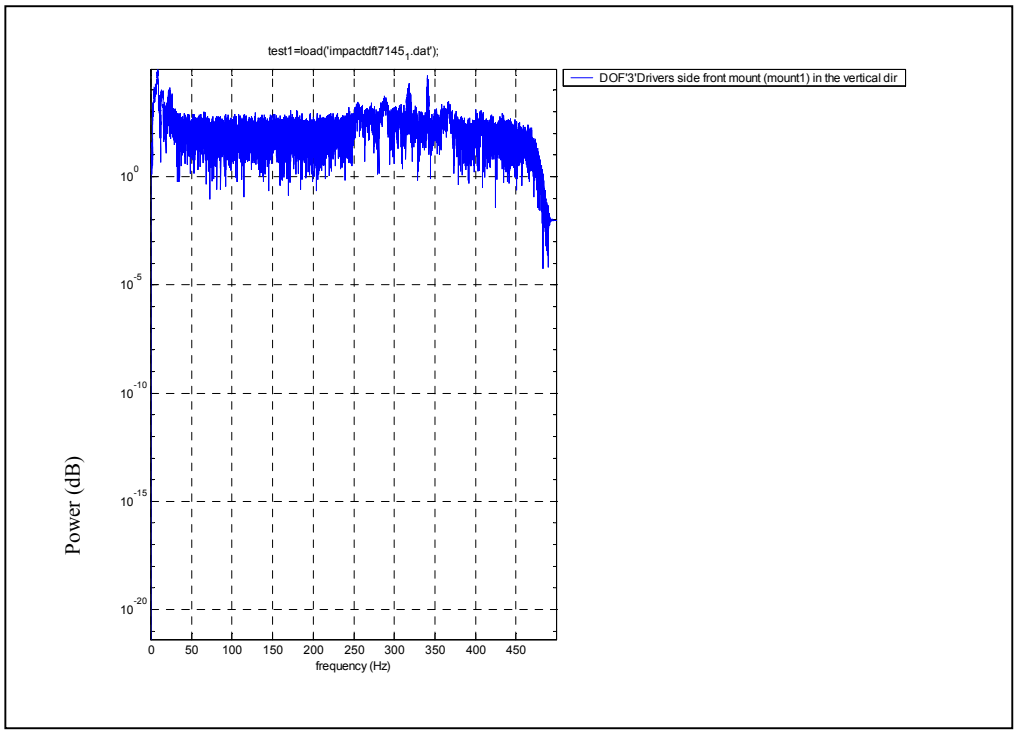


Figure 4-17, DOF3, Drivers side, Front Mount (Mount 1), Vertical Direction.

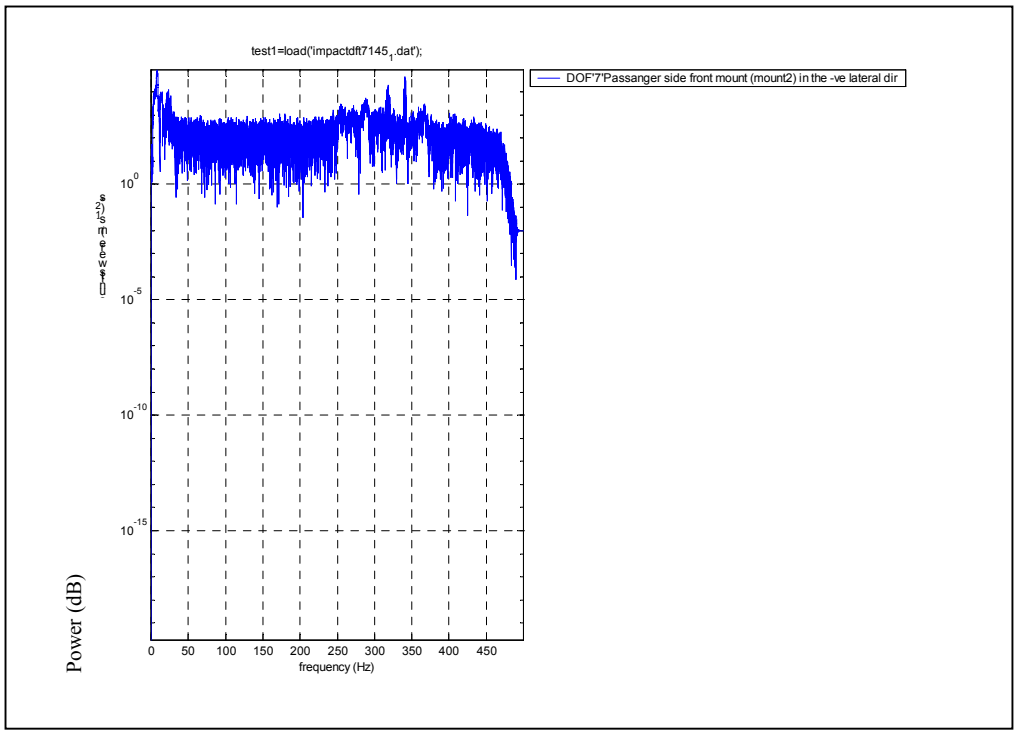


Figure 4-18, DOF7, Passengers side, Front Mount (Mount 2), Lateral Direction.

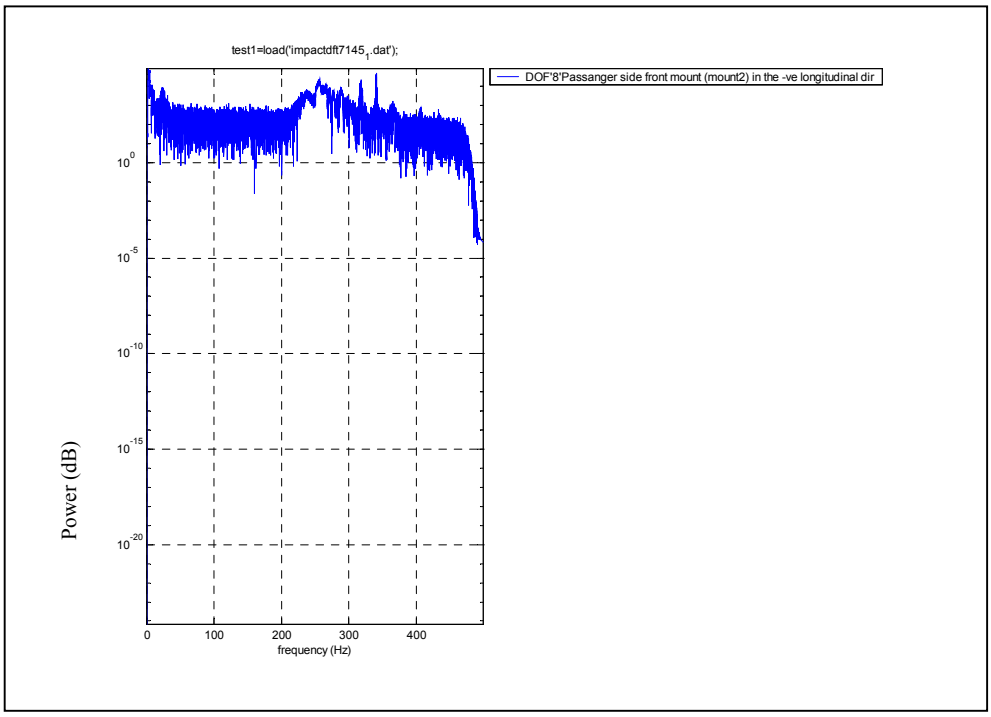


Figure 4-19, DOF8, Passengers side, Front Mount (Mount 2), Longitudinal Direction.

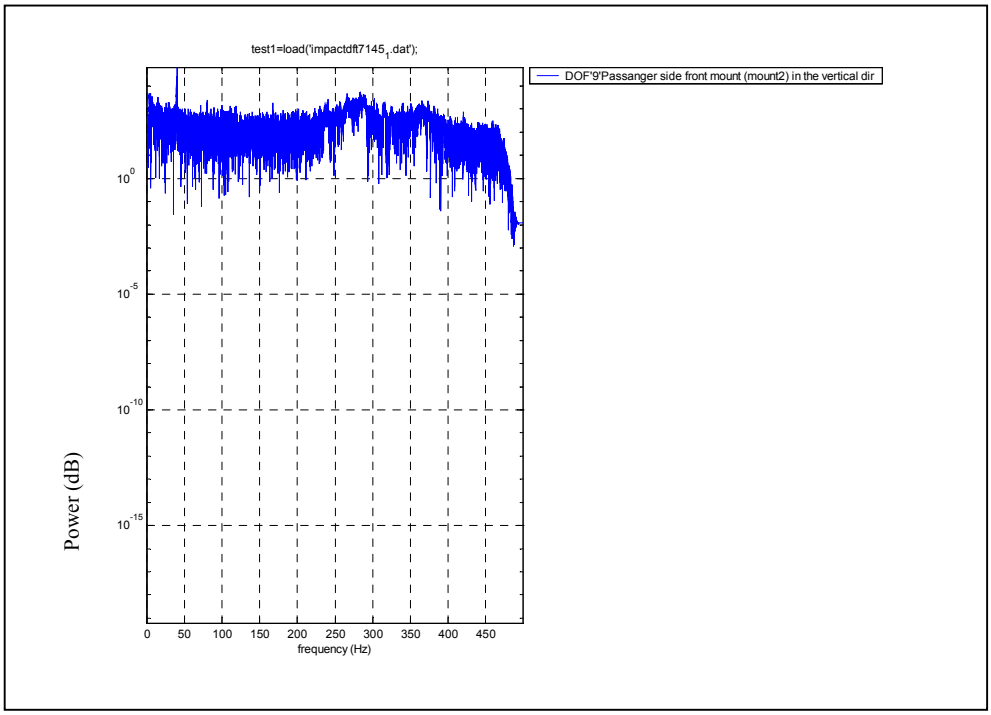


Figure 4-20, DOF9, Passengers side, Front Mount (Mount 2), Vertical Direction.

Table 4-1, Experimental Frequency peaks from the PSD plots in Hz.

f_1=2.2	f_7=4.45	f_{13}=39.9	f_{19}=290
f_2=2.4	f_8=8.3	f_{14}=122	f_{20}=318
f_3=2.41	f_9=8.35	f_{15}=246	f_{21}=340
f_4=2.45	f_{10}=9	f_{16}=255.5	f_{22}=341
f_5=3.6	f_{11}=11	f_{17}=257	f_{23}=367.5
f_6=4	f_{12}=22.5	f_{18}=277	f_{24}=445.3

It would be reasonable to assume that some of the frequencies in Table 4-1 are essentially the same mode with some amount of measurement error. Using data from different tests means an error of ±2 Hz would not be unrealistic, with this test procedure. The natural frequencies of 2.2Hz, 2.4 Hz, 2.41Hz and 2.45Hz were obtained in the lateral direction of all four of the accelerometers at the four mounts.

As this data represents a simple rigid body mode in the lateral direction, these four results have been summarised as one frequency of 2.2Hz. Therefore, combining common frequencies the results are summarised in Table 4-2.

Table 4-2, Summary of Experimental Frequency peaks from the PSD plots in Hz.

f_1=2.2	f_5=9	f_9=122	f_{13}=290	f_{17}=445.3
f_2=3.6	f_6=11	f_{10}=246	f_{14}=318	
f_3=4.45	f_7=22.5	f_{11}=257	f_{15}=341	
f_4=8.35	f_8=39.9	f_{12}=277	f_{16}=367.5	

Many of the lower frequencies, such as f_1 to f_8, can be compared with the mathematical model, which considers the cab of the locomotive as a discrete system of six coordinates. It is worth noting that the higher frequencies, such as f_9 to f_{17}, suggest that there is also continuous system vibration of the beams that are used in the construction of the floor, walls and roof of the cab of the locomotive.

However, as noted in Chapter 2, the only frequencies of interest when considering the whole body vibration effect on the human body are approximately in the range of 1 - 65 Hz. It is the low frequency vibration that has the most power and the relative power of the high frequencies is negligible. This result is demonstrated in Figure 4-15 through to Figure 4-20. The difference between the theoretical results of Table 2-1 and table 4-2, may be due to flexibility of the cab that was neglected in the model.

The only experimental opportunities available were for two locomotives due to the high amount of time that the locomotives are in service. The locomotives measured were DFT 7145 and DFT 7186. This is not to say that these were the only two tests that were conducted, but that every time a test was performed, these were the only two locomotives available at the time. The lack of DFT differing tested made it difficult to draw a conclusion as to whether the frequencies measured are typical of DFT locomotives.

4.6 Continuous System Vibration of the Locomotive Cab.

To see whether the high frequencies, such as f_9 to f_{17} in Table 4-2, were due to structures such as the beams of the floor members, the frequencies were compared to analytical results for beams of the same effective length. The results indicate that the natural frequencies of beams with the lengths in cab construction match the high-energy peaks that are observed in the PSD plots. Hence, the higher natural frequencies may correspond to modes associated with the vibration of the cab structure which points to the need to treat the structure as a continuous system.

Chapter 5

5 Discussion and Analysis of Results

The mathematical model did not give an accurate representation of the vibrations present in the locomotive. However, it did give some indication of the frequencies that would be present. The fact that the experimental results were not able to be produced for different mounting systems was due to the large expense that was involved for Tranz Rail in such a process. Therefore, a complete comparison of the mathematical model with the actual system could not be performed. Had this been possible then the model could have been used to predict the changing of the natural frequencies due to changes of the mounts in the locomotive cab. These results could then have been compared to the experimental natural frequencies obtained for the different mounts. As a result, while a great deal has been discovered and confirmed about DFT cab vibrations a complete analysis was not possible.

5.1 Comparison of Natural Frequencies of Mathematical Model for Differing Mounts.

It is the understanding of the author that Tranz Rail was interested in the use of 'off the shelf items' that could be easily obtained from the Original Equipment Manufacturer (OEM). These items would not have to be custom fabricated or require the fitting of a new mounting system. For this reason inquiries were made about the supplier of the currently used mounts. It appeared that replacement DFT locomotive cab mounts were being supplied by the OEM, the General Motors Corporation Electro-Motive Division Locomotive Group in the United States. However, it was later discovered that these mounts were actually being purchased from the UK mount supplier, Metalastik. After receiving Metalastik catalogues it appeared mounts that were being used are Metalastik standard stock items. The decision was made to try several of the mounting alternatives that were offered and could fit the current

mounting set-up present on the locomotive cabs to see if there was any significant difference in the results. The results in Table 5-1 though Table 5-3 show the mounts used and the resulting frequencies.

Table 5-1, Mount 17-1227-05 (45 Shore).

Static Axial Stiffness = 840 kN/m Dynamic Axial Stiffness = 924 kN/m

Static Radial Stiffness = 2942 kN/m Dynamic Radial Stiffness = 3236 kN/m

Dynamic Magnifier =18

Frequencies (Hz)
$f_1 = 3.6801$
$f_2 = 4.6057$
$f_3 = 5.5911$
$f_4 = 12.91$
$f_5 = 13.763$
$f_6 = 19.371$

Table 5-2, Mount 17-1227-02 (60 Shore).

Static Axial Stiffness = 1575 kN/m Dynamic Axial Stiffness = 1968 kN/m

Static Radial Stiffness = 5512 kN/m Dynamic Radial Stiffness = 6891 kN/m

Dynamic Magnifier = 13

Frequencies (Hz)
$f_1 = 5.37$
$f_2 = 6.7201$
$f_3 = 8.1567$
$f_4 = 18.826$
$f_5 = 20.068$
$f_6 = 28.229$

Table 5-3, Mount 17-1227-03 (70 Shore).

Static Axial Stiffness = 2311 kN/m Dynamic Axial Stiffness = 3120 kN/m

Static Radial Stiffness = 8091 kN/m Dynamic Radial Stiffness = 10921 kN/m

Dynamic Magnifier = 9

Frequencies (Hz)
f_1= 6.759
f_2= 8.457
f_3= 10.262
f_4= 23.662
f_5= 25.214
f_6= 35.427

In these tables it is observed that there is not a significant difference in the range of the natural frequencies for the various mounts. It is also worth noting that the natural frequencies that have been calculated are also within the 1 to 50-80 Hz range, that are arguably detrimental to humans after long periods of exposure.

The relevance of these calculated natural frequencies is questionable when they were compared to the experimentally obtained natural frequencies, for which the mount 17-1227-02 (60 Shore) was used. It can be seen from the lower frequency band of these results in Table 5-4, that the range of frequencies actually covers the complete range of frequencies of the three different mounts in Table 5-1 through to Table 5-3. More specifically, the lowest frequency of Mount 17-1227-05 (45 Shore) is *3.6801 Hz,* and the highest frequency of Mount 17-1227-03 (70 Shore) is *35.427 Hz*. The experimentally determined frequencies that indicated high energy had a range of 2.2Hz to 39.9Hz, much the same frequency range. Hence, these different mounts would not be likely to have a significant impact.

Table 5-4, Lower Frequency band of high-energy frequencies, experimentally determined in Hz.

Frequency (Hz)
f_1=2.2
f_2=3.6
f_3=4.45
f_4=8.35
f_5=9
f_6=11
f_7=22.5
f_8=39.9

5.2 Steady-State Response of the DFT Locomotive

An additional reason for questioning the effectiveness of the mount or the relevance of the natural frequencies of the cab mounts was due to the results of in-service measurements that were taken. The results showed that the power of driving frequencies being transmitted through the mounts was so high that literally none of the characteristics of the system were detectable[1]. The speed of the engine used was approximately 900rpm or 94.24rads^{-1}, 15rps or 15Hz.

With the accelerometers mounted as documented in Chapter 4, a series of tests of the cab of the locomotive were conducted. The locomotive controls allow the selection of 8 different active preset power levels. Accelerations of the locomotive in all of the different power notch settings at various speeds were obtained and stored. The speeds that these tests were conducted at are 20 km/hr, 40 km/hr, 60 km/hr and 80km/hr.

[1] That is the isolated cab of the locomotive

In all cases there was one dominant high energy peak that swamped any other high energy peaks present. This peak was always related to the power notch of the locomotive regardless of the speed. Three of eight plots of the PSD can be seen in Figure 5-1 through to Figure 5-3 for Degree Of Freedom 1 (DOF1), the mount at the front on the drivers side, in the lateral direction.

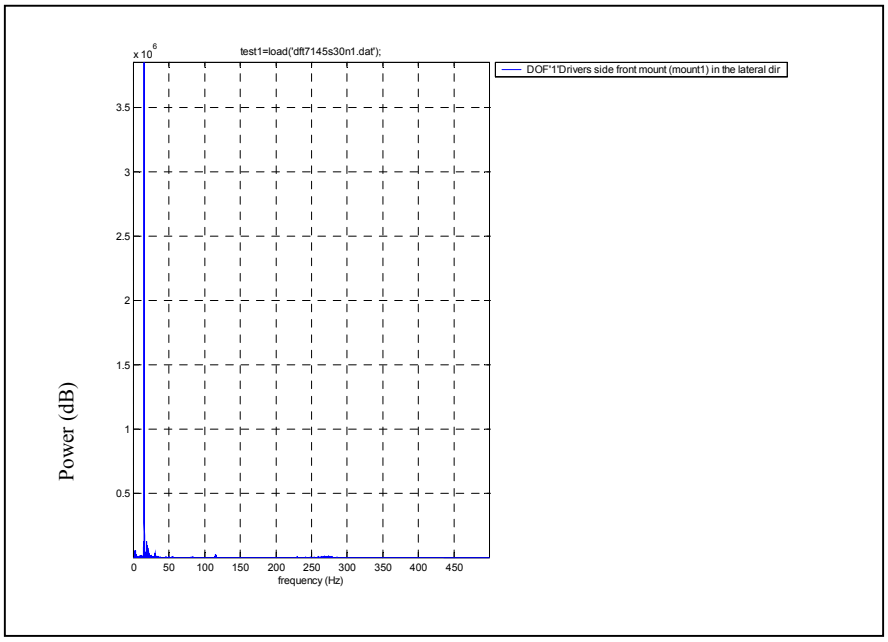


Figure 5-1, Power observed from the PSD for DFT7145, at 30km/hr in Notch 1.

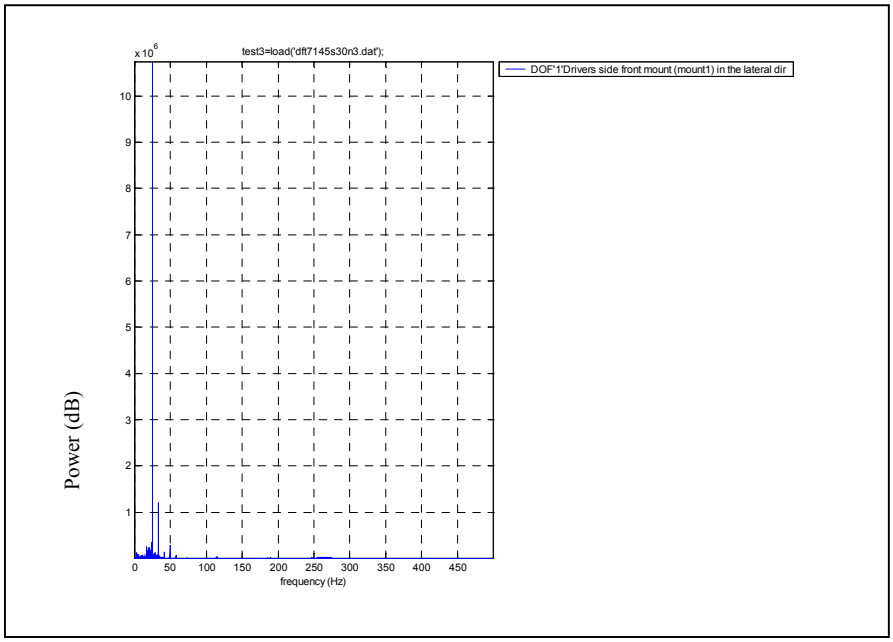


Figure 5-2, Power observed from the PSD for DFT7145, at 30km/hr in Notch 3.

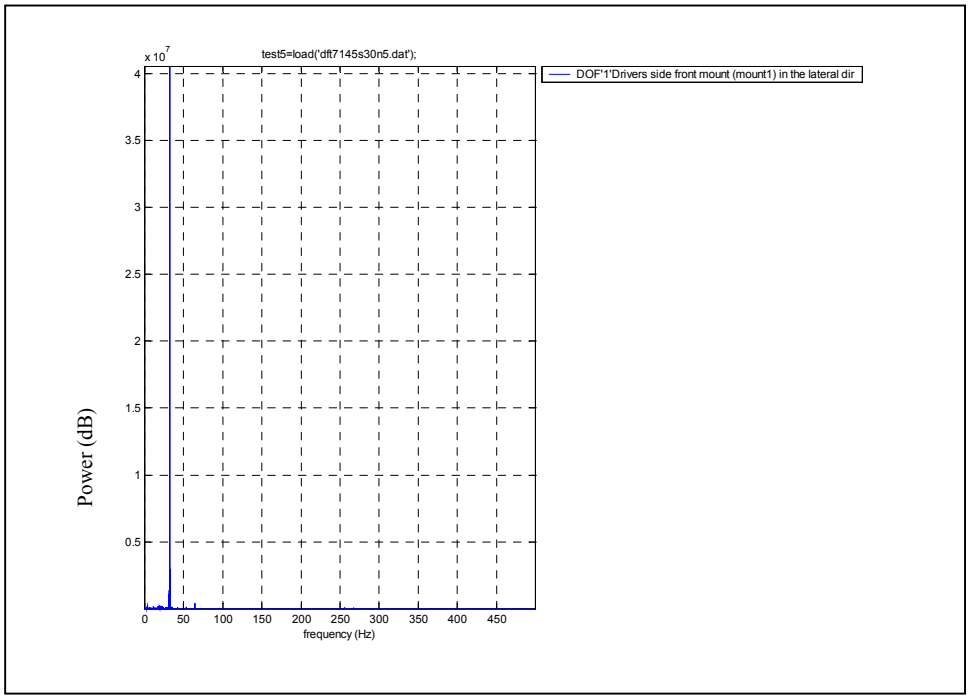


Figure 5-3, Power observed from the PSD for DFT7145, at 30km/hr in Notch 5.

The idea behind performing these tests was to be able to pick out the natural frequencies of the cab of the locomotive by overlaying the PSD of each of the tests. That is to overlay all of the PSD plots for the locomotive at 20km/hr at all of the 8 power notch settings. Similarly all 8 power-notch settings measured for the locomotive at 40 km/hr, and so on, up to the 80 km/hr measurements recorded. As an analogy to an auto-correlation, a sample of the combined results of these graphs when they are overlaid is shown in Figure 5-4 for DOF1.

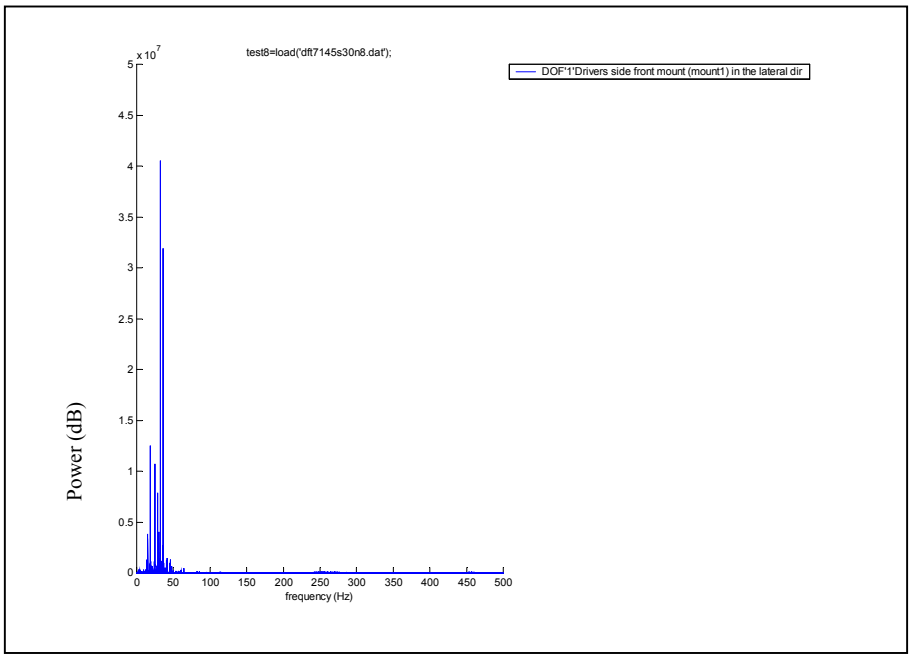


Figure 5-4, The eight combined notches for DFT7145 at 30 km/hr.

Due to the very large number of graphs necessary to represent the data the maximum peak for each notch at each speed was recorded to summarise the peaks observed. Figure 5-5 was constructed for each set of tests conducted at a set speed.

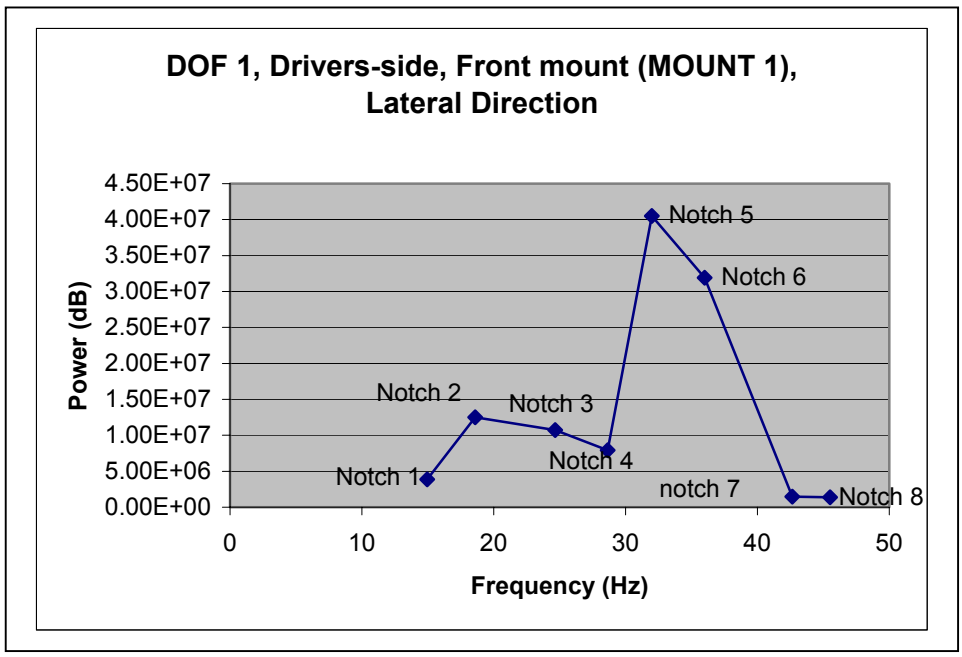


Figure 5-5 Summary of the Power, at different notches and the frequencies at which they occur for the Lateral direction at all mounts.

Figure 5-5 represents the lateral acceleration of Mount 1. However, it was interesting to note that although the actual figures varied, the general shape of this graph is the same for all four mounts for the lateral direction. Likewise, it was found that the vibration occurring at each mount for the longitudinal direction gave a distinctive shape different from that of the lateral direction. The longitudinal data is shown in Figure 5-6 and the vertical data in Figure 5-7.

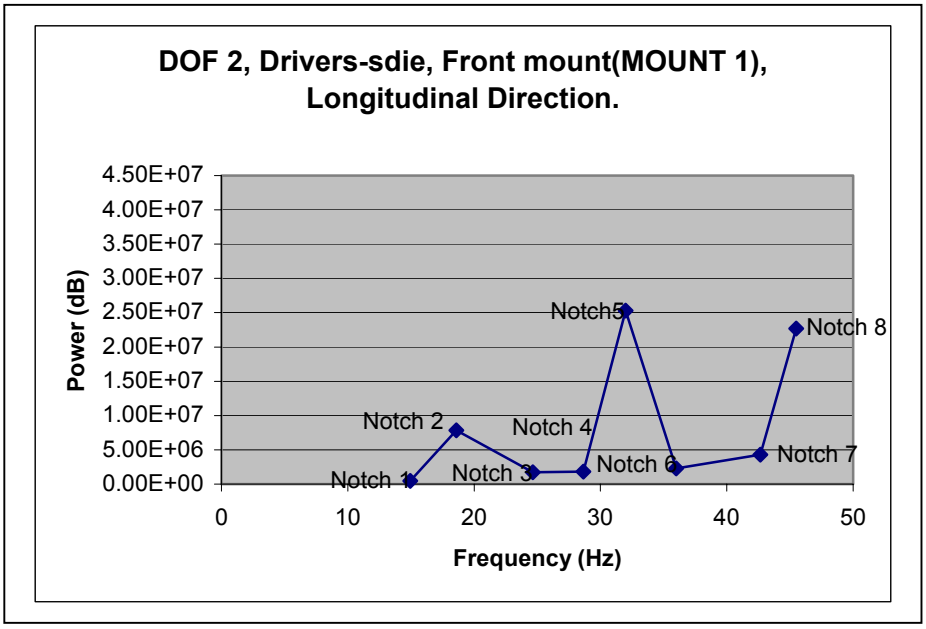


Figure 5-6, Summary of the Power, at different notches and the frequencies at which they occur for the Longitudinal direction at all mounts.

The fact that all four mounts indicated similar measured values for the power at the different notches and frequencies indicates that the vibrational input through the mounts meets rigid body assumptions.

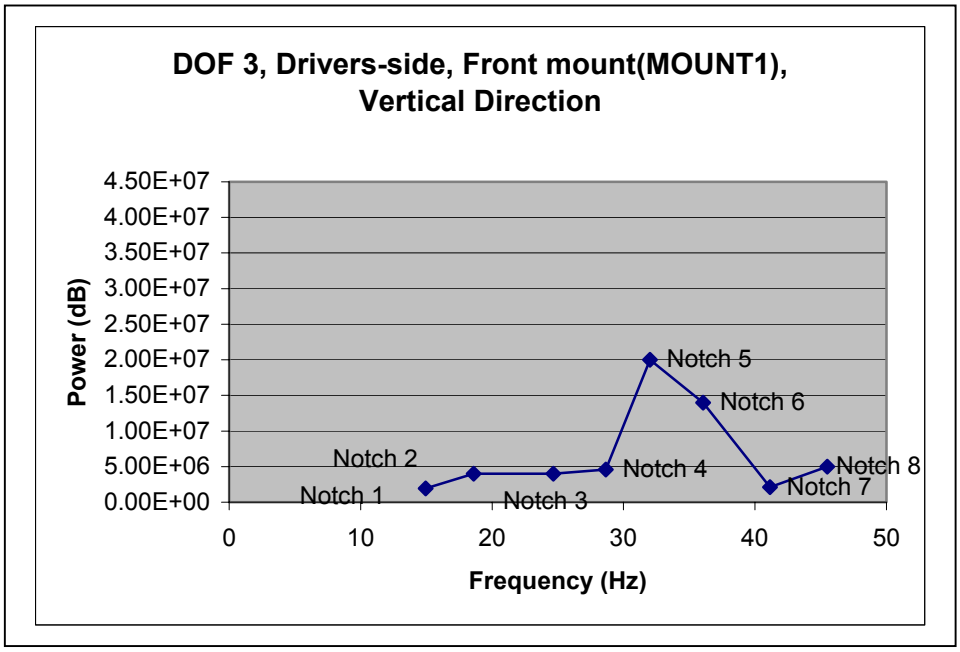


Figure 5-7, Summary of the Power, at different notches and the frequencies at which they occur for the vertical direction at all mounts.

These 'Power Summaries' were also compiled for the lateral, longitudinal and vertical acceleration present at the drivers seat. Here again it was found that for each test there was a distinct shape to the graphs for the lateral, longitudinal and vertical acceleration. However, these were different to the lateral, longitudinal and vertical acceleration plots for the four mounts. The drivers seat data is presented in

Figure **5-8** through Figure 5-10. The difference between these plots and the plots of the four mounts indicates that there is some continuous system vibration of the beam members of the floor that the mathematical system model cannot account for, as well as some possible filtering due to dynamic modes of the seat structure.

The summary of the driving frequencies for the mounts of the locomotive cab mounts, show that the high driving frequencies transferred from the locomotive to the locomotive cab are above the natural frequencies of the cab of the locomotive. Unfortunately, the driving frequencies present in the locomotive have so much power associated with them that the position of the natural frequencies of the cab of the locomotive due to the mounts that are currently used on the cab is irrelevant. The RMS accelerations for each point were also calculated and enabled the use of plotting

the RMS accelerations with their respective frequencies. Figure 5-11 plots the RMS acceleration at the drivers seat for each notch consecutively.

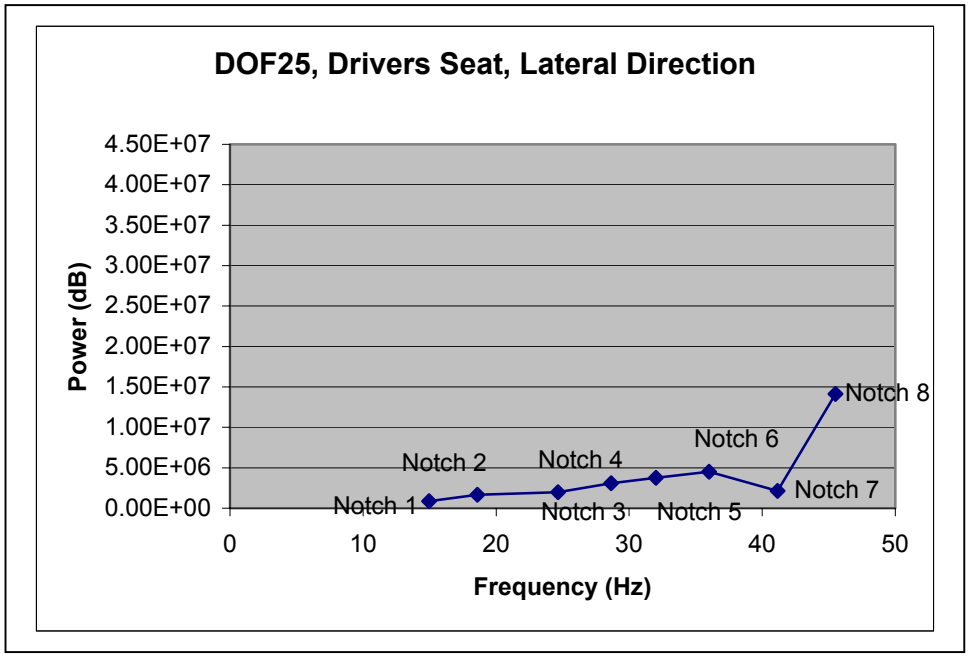


Figure 5-8, Summary of the Power, at different notches and the frequencies at which they occur for the lateral direction at the drivers seat.

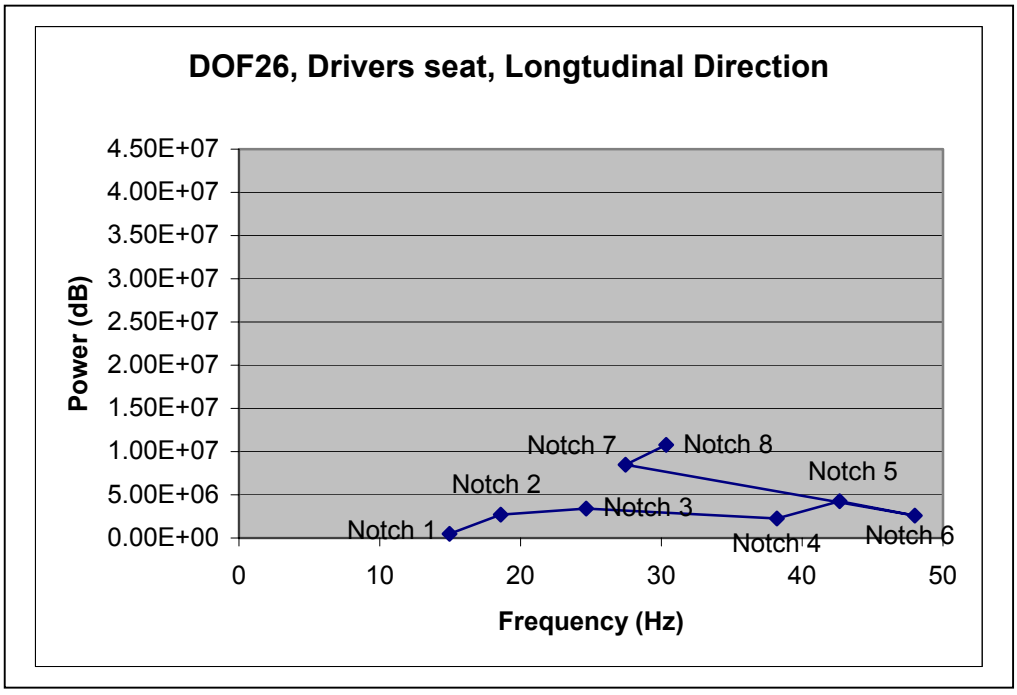


Figure 5-9, Summary of the Power, at different notches and the frequencies at which they occur for the lateral direction at the drivers seat.

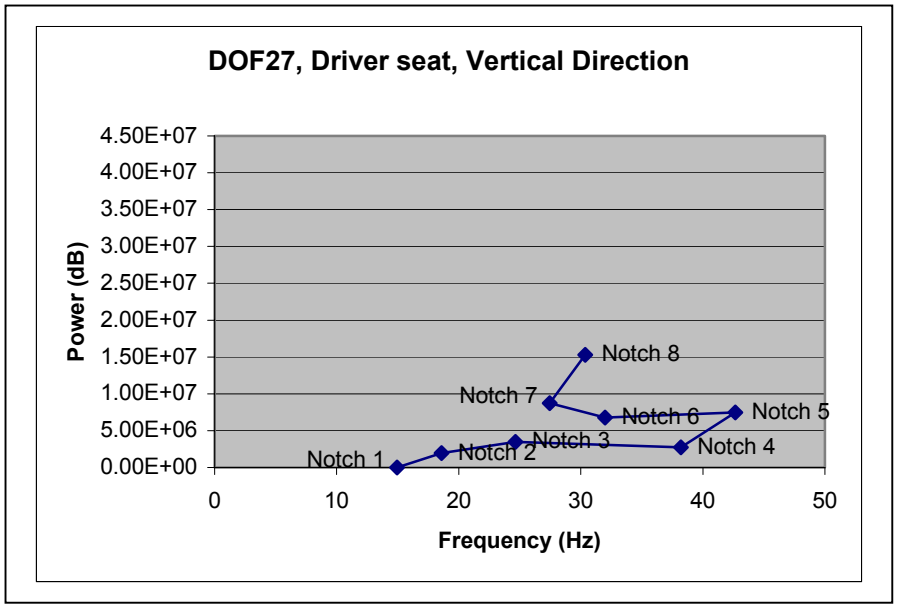


Figure 5-10, Summary of the Power, at different notches, and the frequencies at which they occur for the lateral direction at the drivers seat.

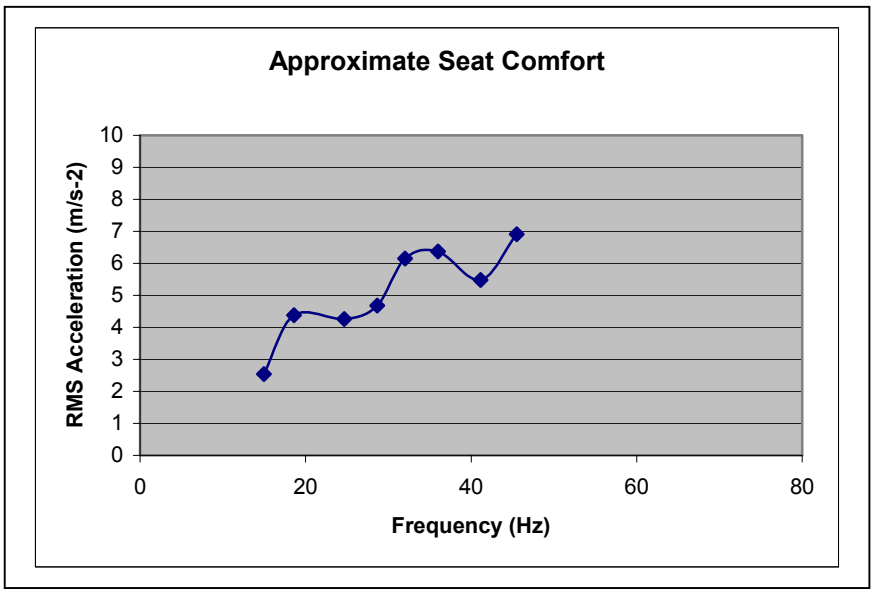


Figure 5-11, RMS Acceleration of the drivers seat.

The calculation for the RMS acceleration is based on the basic comfort formulation of ORE [1989], which is based on ISO 2631 [1990a,1990b,1990c].

$$N_{RMScomfort} = 6\sqrt{(a_{x\_RMS})^2 + (a_{y\_RMS})^2 + (a_{z\_RMS})^2}$$

Equation (5-1)

A generally acceptable standard for the comparison of RMS acceleration, with the expected levels of comfort is also derived from ISO 2631 [1990a,1990b,1990c] and Wilde [1980] as shown in Figure 5-12.

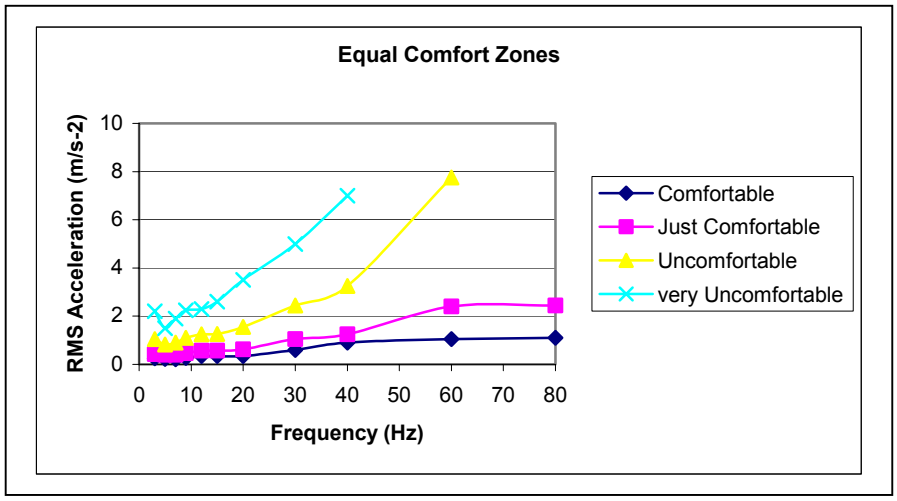


Figure 5-12, Comfort index of RMS acceleration for Railway application.

When the results of Figure 5-11 are overlaid on Figure 5-12, it is found that in the frequency range of 15-36Hz the approximate RMS acceleration measured at the base of the drivers seat were above the 'very uncomfortable' line for durations longer that 30 minutes. It is also found that the frequencies above 36 Hz were in the 'uncomfortable range'.

The data that was used to compile Figure 5-12 was based on a nominal half an hour journey. Where the occupants of the locomotive cab are in the cab for extended periods of time, such as a normal work shift, there is no data to be able to make such a

comparison except to say that discomfort experienced by the acceleration would likely be worse for extended periods at these levels.

Chapter 6

6 Conclusions and Recommendations

6.1 Conclusion

A mathematical model was developed taking the locomotive cab to be a rigid body consisting of 6 Degrees Of Freedom supported on a rigid underframe. It was able to give a solution within the range of the lower frequencies present. The numerical results generated did not agree with the experimental results, possible this difference may be due to the flexibility of the cab, and would need to be investigated. This model was inadequate for the selection of another elastomeric mounting alternative. The discrete system natural frequencies of the locomotive cab were calculated for several of the mounts available now but the experimental frequencies obtained for the current mount covered all of the extremes of these calculated natural frequency ranges. The variation of the theoretical natural frequencies of the different mount types may give an indication of the difference of the experimentally determined natural frequencies for the different mount types if a different mounting alternative is used.

An experimental set-up consisting of easily mountable accelerometer units and data acquisition system was designed and commissioned. Measure results indicate that the mountings currently used do not effectively isolate the cab, and in some case amplify vibration energy. To test the rigidity of the cab, accelerations along all three translational directions of each mount were measured and then six of the measurements were used to predict another six accelerations. These results showed that although the estimate of the other coordinates was for some of the coordinates within a reasonable percentage error, the percentage error calculated in the estimation was too high to be of real use. This

factor also pointed to potential non-linearity in the system and to continuous vibration in the floor members of the cab of the locomotive not modeled by the original model.

Auto-correlation results from acceleration measurements at the base of the mount were overlaid with the cross-correlation of the acceleration measurements of the cab side of the mounts. Theses results showed that the current mounts have 100% transmissibility. In some cases and directions the mount was shown to actually amplify the vibrational input from the locomotive.

The amplitude of the steady-state vibration of the locomotive cab that was being transmitted was so great that it rendered the dynamic characteristics of the cab insignificant.

While elastomeric mounts may be the most effective form of isolation of the cab, the test results indicate that these mounts could be far better optimized.

Several alternatives of viable elastomeric mounts from the Metalstik Manufactures catalogue were used to calculate the theoretical natural frequencies of the cab. Only mounts that would fit easily into the existing fixture were considered. However, from the experimental results, the elastomeric mounts were incapable of attenuating the vibration. Additionally, the vibration levels due to locomotive running were found to be well above accepted standards for comfort. Finally, while the rigid body assumption was shown valid it was found that there are other continuous vibrations and non-linear effects that are also very important in determining cab vibration.

6.2 Recommendations

For current cabs a far safer, practical mount is recommended to reduce transmissibility and minimize vibration at the drivers seat to a level below the specified ranges for

discomfort found in ISO and other specifications[Wilde,1980]. When considering the changing of the mounts, the strength of the mount must also be checked. With the advances in the electronics industry a feasibility study could be performed to see if active control of the vibration of the cab mount is a viable alternative as it may be able to provide better overall isolation of the locomotive cab. Further research and development to provide more effective isolation should be continued including such areas as active control. An investigation of the use of a continuous system model for the modeling of the floor, walls and roof of the locomotive cab would be useful in order to get a more accurate approximation for the natural frequencies of the locomotive cab.

Bibliography

Adams, O., "*Elements Of Diesel Engineering*", New York, Norman W. Henley Publishing Co., 1937.

Ahmadian, M., "*APPLICATION OF IMPEDENCE METHOD FOR PREDICTING VIBRATION OF A LOCOMOTIVE CAB*", Rail Transportation, RTD-Vol.15, pp.45-52, ASME, 1995.

Australian Standard, "*AS 2670.1-1990, ISO 2631/1-1985, Evaluation of human exposure to whole-body vibration, Part 1: General requirements*", Sydney, Standards Australia, 1990.

Australian Standard, "*AS 2670.2-1990, ISO 2631/2-1989, Evaluation of human exposure to whole-body vibration, Part 2: Continuous and shock induced vibration in buildings (1 to 80 Hz)*", Sydney, Standards Australia, 1990.

Australian Standard, "*AS 2670.3-1990, ISO 2631/3-1985, Evaluation of human exposure to whole-body vibration, Part 3: Evaluation of exposure to whole-body z-axis vertical vibration in the frequency range 0.1 to 0.63Hz*", Sydney, Standards Australia, 1990.

Beards, C. F., "*Vibrations and Control Systems*", England, Ellis Horwood Ltd., 1988.

Beards, C. F., "*Engineering Vibration Analysis With Application To Control Systems*", England, J.W. Arrowsmith Ltd., 1995.

British Standard Institution, "*BS 3015:1991, ISO 2041:1990, Glossary of terms to mechanical vibration and shock*", London, BSI Publications, 1990.

British Standard Institution, "*BS ISO 5348:1998, Mechanical vibration and shock- Mechanical mounting of accelerometers*", 2nd Ed., London, BSI Publications, 1998.

British Standard Institution, "*BS ISO 5805:1997, Mechanical vibration and shock- Human exposure- Vocabulary*", London, BSI Publications, 1997.

British Standard Institution, "*BS 6794:1986, ISO 8002-1986, Reporting measured vibration data for land vehicles*", London, BSI Publications, 1986.

British Standard Institution, "*BS 6955: Part 1: 1994: ISO 5347-1: 1993, Calibration of vibration and shock pick-ups, Part 1. Methods for primary vibration calibration by laser interferometry*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 2: 1994: ISO 5347-2: 1993, Calibration of vibration and shock pick-ups, Part 2. Methods for primary shock calibration by light cutting*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 3: 1994: ISO 5347-3: 1993, Calibration of vibration and shock pick-ups, Part 3. Methods for secondary vibration calibration*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 4: 1994: ISO 5347-4: 1993, Calibration of vibration and shock pick-ups, Part 4. Methods for secondary shock calibration*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 5: 1994: ISO 5347-5: 1993, Calibration of vibration and shock pick-ups, Part 5. Methods for calibration by earth's gravitation*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 6: 1994: ISO 5347-6: 1993, Calibration of vibration and shock pick-ups, Part 6. Methods for primary vibration calibration at low frequencies*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 10: 1994: ISO 5347-10: 1993, Calibration of vibration and shock pick-ups, Part 10. Methods for primary calibration by high impact shocks*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 14: 1994: ISO 5347-14: 1993, Calibration of vibration and shock pick-ups, Part 14. Methods of test for*

resonance frequency of undamped accelerometers on a steel block", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 15: 1994: ISO 5347-15: 1993, Calibration of vibration and shock pick-ups, Part 15. Methods of test for acoustic sensitivity*", London, BSI Publications, 1994.

British Standard Institution, "*BS 6955: Part 20: 1997: ISO 5347-20: 1997, Calibration of vibration and shock pick-ups, Part 20. Primary vibration calibration by the reciprocity method*", London, BSI Publications, 1997.

British Standard Institution, "*BS 6955: Part 22: 1997: ISO 5347-22: 1997, Calibration of vibration and shock pick-ups, Part 22. Acceleration resonance testing- General methods*", London, BSI Publications, 1997.

British Standard Institution, "*DD ENV 28041:1993 ISO 8041:1990, Human response to vibration measuring instrumentation*", London, BSI Publications, 1990.

Broch, J. T., "*Mechanical Vibration And Shock Measurements, 'Brüel & Kjær's series of books on the fundamental of physical measurement and analysis'* ", 2nd Ed., Denmark, K. Larson & Son, 1980.

Buchholdt, H., "*Structural Dynamics for Engineers*", London, Thomas Telford Publications, 1997.

Butsuen, T., *"Application of Direct System Identification Method for Engine Rigid Body Mount System"*, Society of Automotive Engineers, UK, 3.525, 860551, 1987.

Chen, Y., "*Vibrations: Theoretical Methods*", USA, Addison-Wesley Publishing Company, Inc., 1966.

Clough, R.W., Penzien, J., "*Dynamics of Structures*", New York, McGraw-Hill Book Company, 1975.

Crandall, S. H., "*Dynamics Of Mechanical And Electromechanical Systems*", New York, McGraw-Hill Book Company, 1968.

Crede, C. E., Harris, C. M., "*Shock And Vibration Handbook 'vol 1: Basic Theory And Measurements'* ", New York, M^{c}Graw-Hill Book Company Inc., vol.1, 1961.

Crede, C. E., Harris, C. M., "*Shock And Vibration Handbook 'vol 2: Data Analysis, Testing, And Methods of Control'* ", New York, M^{c}Graw-Hill Book Company Inc., vol.2, 1961.

Crede, C. E., Harris, C. M., "*Shock And Vibration Handbook 'vol 3: Engineering Design And Environmental Conditions'* ", New York, M^{c}Graw-Hill Book Company Inc., vol.3, 1961.

Dhruna, C. J., "*Experimentally-Based Analytical Prediction of Structural Vibration*", Virginia, Masters Thesis, Virginia Polytechnic Institute and State University, 1997.

Dukkipati, R. V., "*Computer-Aided Simulation in Railway Dynamics*", New York, Marcel Dekker inc., 1988.

Foss, K. A., *"Co-Ordinates Which Uncouple the Equations of Motion of Damped Linear Dynamic Systems"*, Journal of Applied Mechanics, 25, 1958.

Garner, J. A., "*The Development of a Model for Organisational Integration Through Integrated Hypothesis Development*", Ph.D. Thesis, University of Canterbury Mechanical Engineering Department, 1991.

Goddard, David. W., "*An Integrated Computer Aided Design System for Railway Freight Vehicles*", Ph.D. Thesis, University of Canterbury Mechanical Engineering Department, 1987.

Grootenhuis, P., Ewins, D.J., "*Vibration of a Spring-Supported Body*", Journal Mechanical Engineering Science, vol2, No 2, 1965.

Hagino, Y., Furuishi, Y., Makigawa, Y., *"Active Control for Body Vibration of F.W.D. Car"*, Society of Automotive Engineers, UK, 3.534, 860552, 1987.

Hodgetts, D., McDonald, A. M., *"The Transmission of Piston Forces to the Mounts of an Engine"*, Cranfield Institute of Technology, IMechE, 1979.

IMechE Conference Transactions 1983-7. "*Proceeding Of The Institution Of Mechanical Engineers, International Conference 'Diesel Locomotives For The Future'* ", London, Mechanical Engineering Publications Ltd., 1987.

IMechE Conference Transactions 1991-7. "*Third International Conference on 'Train Maintenance Tomorrow…And Beyond'* ", London, Mechanical Engineering Publications Ltd., 1997.

Iwnicki, S. D., Brickle, B. V., *"The Dynamic Behaviour of a Rail Locomotive with Solid Rubber Tyres and Flange Steel Wheels"*, The dynamics of Vehicles on road and tracks, Supplement to Vehicle System Dynamics, vol 17, 1988.

Kelley, A., Pohl, I., "*A Book on C: Programming in C*", 3rd Ed., Harlow, England, Addison-Wesley, 1995.

Korablev, S., Shapin, V., Filatov, Y., "*Vibration Diagnostics In Precision Instruments*", New York, Hemisphere Publishing Corporation, 1989.

Kortum, W., *"Review of Multibody Computer Codes of Vehicle System Dynamics"*, Supplement to Vehicle System Dynamics, vol 22, 1992.

Kotlin, J.J., IMechE Conference Publications 1982-7, "*The GM/EMD model F3A diesel engine for rail application*", London, The Next Generation of Diesel Engines For Rail Traction, C87/82, pp.69-80, Mechanical Engineering Publications Ltd., 1982.

Knall, V., "*Railway noise and vibration: Effects and criteria*", Great Britain, Journal of Sound and Vibrations, Vol. 193, No. 1, pp. 9-20, 1996.

Li, D. B., Yam, L. H., "*Modal Synthesis Method for Vibration Isolation Design of Massive Rotating Machines Resiliently Supported by an Elastic Structure*", Journal of Sound and Vibrations, 231(1), 2000.

Lyle Cummins, C. Jr., "*Diesel's Engine*", Oregon, dba Carnot Press, vol.1, 1993.

McCallion, H., "*Vibration of Linear Mechanical Systems*", London, Longman Group Ltd, 1973.

Meirovitch, L., "*Analytical Method in Vibrations*", New York, Macmillan Company, 1967.

Meirovitch, L., "*Computational Methods In Structural Dynamics*", Netherlands, Sijthoff & Noordhoff International Publishers B.V., 1980.

Meirovitch, L., "*Elements Of Vibration Analysis*", New York, M^{c}Graw-Hill Book Company Inc., 1986.

Office for Research and Experiments of the International Union of Railways, *"Question B 153, Application of the ISO 2631 standard to Railway vehicles"*, UTRECHT (Netherlands), 1989.

Porter, S. R. M., "*The Mechanics Of A Locomotive On Curved Track*", London, The Railway Gazette, 1935.

Redfield, R. C., Karnopp, D. C., *"Optimal Performance of Variable Component Suspensions"*, Vehicle System Dynamics, number 5, vol 17, 1988.

Ringheim, M., "*Fifth international workshop on tracked transit system noise 1995: A summary of conclusions*", Great Britain, Journal of Sound and Vibrations, Vol. 193, No. 1, pp. 3-7, 1996.

Shoureshi, R., Graf, P. L., *"Adaptive Hydraulic Engine Mounts"*, Society of Automotive Engineers, UK, 3.516, 860549, 1987.

Srinivasan, P., "*Nonlinear Mechanical Vibrations*", India, New Age International (P) Ltd., 1995.

Steidel, R. F. Jr., "*An Introduction to Mechanical Vibrations*", 3rd Ed., New York, John Wiley & Sons., 1989

Thomson, W. T., "*Theory of Vibration With Applications*", London, George Allen & Unwin Inc., 1981.

Tse, F.S., Morse, I.E., Hinkle, R.T., "*Mechanical Vibrations, Theory And Applications*", 2nd Ed., Boston, Allyn & Bacon Inc., 1978.

Venezia, J. J. Jr., "*Vibration Modeling and Experimental Analysis of a Locomotive Cab*", Virginia, Masters Thesis, Virginia Polytechnic Institute and State University, 1997.

Wilde, G.J.S., Stinson, J.F., "*Human factors considerations in locomotive cab design*", Canada, Canadian Institute of Guided Ground Transport Queen's University at Kingston, Ontario, CIGGT Report No. 80-9, pp.63-91, 1980.

Appendix A: DF to DFT conversion.

Comparison between DF and DFT Locomotives

TRL Loco Type	DF	DFT
EMD designation	GL22MC	GT22MC
Road numbers		
Series 1 1979 20 No	6006-6202	7117-7335
Series 2 1981 10 No	6219-6317	7008-7104
Maximum Height	3661 mm	3815 mm
Maximum width	2667 mm	2667 mm
Total Weight	85,700 kg	86,400 kg
Axle Weight	14,300 kg	14,400 kg
Ballasting	0 kg	600 kg
Increase	0 kg	700 kg
Max.Tract.Eff.@ 25 % adhesion	210 kN	210 kN
Engine Type	EMD 12-645E	EMD 12-645E3C
Tractive Horsepower	1500 hp	2100 hp
Auxiliary horsepower	155 hp	196 hp
Engine horsepower	1650 hp	2475 hp
Gear Ratio	57/16	57/16
Maximum Operating Speed	113 kph	113 kph
Passenger Speed	100 kph	100 kph
Freight Speed	80 kph	80 kph
Minimum Continuous Speed (Notch 8)	16 kph	26 kph
Short Time Current Ratings		
Continuous	450 Amps	450 Amps
1 hour	485 Amps	485 Amps
1/2 hour	510 Amps	510 Amps
1/4 hour	545 Amps	545 Amps
$ Overhaul (DF standard, DFT modified)	$400,000 (std ovhl)	$900,000 (prototype)

Comparison runs between Christchurch and Picton (350 km)

	DF	DFT	Improve	Percent
Time Elapsed minutes (Actual)	450	376	74	16 %
Fuel Used litres (Actual)	2050	1790	260	12 %
Train Weight tonnes (Actual)	785	860	75	9 %

DFT Locomotives

The following document outlines the modifications required to convert a DF locomotive to DFT specification.

History

A prototype, DFT7008, was constructed between September and November 1992, and run during 1993 as a trial. A derailment and a number of small faults unrelated to the modifications sidelined the locomotive for about two months in total. The decision was made to modify 10 more DF's to DFT specification. These where carried out at Transtec Engineering Hutt, between February and September 1994. Two further batches of locomotives where converted between October 1995 and January 1997.

Overview

While being shopped for a G2 overhaul at 1.2 million kilometres, The DF's will have extra modifications carried out to bring them up to DFT specification. These modifications were designed by General Motors Diesel Division using standard EMD parts. The locomotive would then exit the shops as a DFT.

The major change is to replace the engine with the latest 645E3C turbo version of the EMD V12-645 2 stroke. As the engine will produce 50 % more power it will produce more heat and require more intake air. Subsequently the cooling systems, both oil and water, and the intake air volume will have to be increased.

ENGINE	Engine converted from 645E to 645E3C specification
EQUIPMENT BLOWER	Replaced and relocated to left side of locomotive.
TRACTION MOTOR DUCT	Modified to line up with relocated blower
INTAKE AIR FILTER	Replaced to line up with turbocharger, extra capacity
INTAKE AIR	Inertial filter on one side replaced with bigger unit.
EXHAUST BULGE	Longhood modified to clear taller exhaust manifold.
EQUIPMENT RACK	New equipment rack replaces old, behind engine.
COMPRESSOR RELOCATED	Compressor moved back 159mm to clear new rack.
RADIATORS & PIPING	New radiator, cooling system piping & structural mods.
ELECTRICAL	3 control modules replaced and soakback pump wiring.
BALLASTING	600kg installed in shorthood to balance end for end.

The modifications will bring the loco up to GT22MC specification from GL22MC. This is a standard configuration as used by several Railroads, thus the components are all well proven in this combination.

Improvements

The DFT locomotive will travel approximately 15 kph faster than a DF against the same train load. It will use 10% less fuel and depending on the journey save 20 minutes per 100km travelled.

Engine

The Engine will be re-built from a 645E type to 645E3C. The main components used from the old engine are the crankcase, crankshaft, oil pan and flywheel.

The block used must be manufactured after 1972 and be of a 645E type.

The new engine will have the two Roots blowers replaced with the latest version of the EMD Turbocharger. This turbocharger is a two stage design, being mechanically driven by the camshaft drive gears at low engine speeds and exhaust gas driven at high engine speeds. This is accomplished by a mechanical clutch that allows the exhaust driven turbine to take over, as its rotating speed exceeds that of the drive gears. The exhaust turbine is always assisting the mechanical drive but does not fully take over until the engine is in notches 7 and 8.

The turbocharger will produce more inlet air pressure than the Roots blown engine and will be more efficient. The improvement in Brake Specific Fuel Consumption, ie the mass (kg) of fuel burnt per kW of output power, is on average 4 % better and in notch 8 it is 14% better.

The fuel injectors will be replaced with higher output types for the increased fuel demand.

The compression ratio remains at 16:1

The turbocharger configuration will include a new camshaft drive gear train, aftercoolers, air ducts, eductor and lube oil separator, exhaust manifold and heat shields.
The aftercoolers are water cooled radiators between the turbocharger and the engine air box inlets. These cool the incoming inlet air, thus reducing its effective volume and allowing higher charge density of air.

The new exhaust section has manifold heat shields and an inbuilt screen to protect the turbo from debris.

Turbo access is limited and special lifting gear will be required to manoeuvre the aftercooler cores and turbocharger into place on the engine.
There is access to top of the Turbo to observe its run down time. This will show a tightening up of bearings etc.

The camshaft drive gears drive the camshafts, turbo and the auxiliary generator so are changed for the new configuration. The housing is also changed and incorporates an oil filter for the turbocharger.

The power assemblies will be replaced with General Motors new Premium Power assemblies (Diamond 5). These power assemblies incorporate redesigned heads to prevent fireface cracking, rocking type piston pins, laser hardened upper bores (HUB) on the cylinder liner, resistance hardened #1 piston ring grooves, tin plated piston skirts and prestressed stainless steel piston rings. These improvements should enable the Premium power assembly to last for a theoretical 1 million miles (1.6 million kilometres) between changes as opposed to the present life of 600,000 kilometres.

The oil pan will be modified to accept the pick-up and mounting for the new turbo soak back pump, filters and relay.

The soakback pump is the pre and post lubrication pump for the Turbocharger.
At start-up the soakback pump pre-lubes the turbo for 35 minutes once the engine control switch has been switched to prime. Once the engine is running a control valve changes the supply to full engine oil pressure. After the engine is shut down the soakback pump maintains oil pressure to the turbo for 35 minutes. This lubricates the turbo bearings while it is spinning down and it also protects the turbo from heat soak after the engine stops.

The new engine components are the turbocharger, turbocharger drive gears and stub shafts, soak back pump , soak back pump filter and relays, camshafts and rollers, power assemblies, water pumps, lube oil pumps, power assembly crab bolts and crabs, head frames, governor, injectors, water pumps, oil pumps, exhaust section, aftercoolers, and exhaust manifolds.

The engine will feature new power assembly plate crabs which are made in one piece between pairs of cylinders.

The crab bolts and nuts are of the latest design. These feature reduced diameter bolts and higher torque. This leads to extra bolt stretch thus avoids re-torquing and has less leaks.

The induction end (back end) of the engine will require 12 extra holes drilled and tapped for the new aftercooler manifolds.

The air box handhole covers should not be of the cast type due to increased air box pressures.

The head frame needs replacing due to cut outs in end for old roots blowers and changes in the internal oil system. (Reused the old ones later on)

An extra oil pressure switch will be added to the turbo oil supply and will not allow the engine to start unless there is adequate oil pressure.

There is a new engine protection device (EPD). This measures the pressure differential across the water pumps rather than absolute pressure at one point. This gives a better indication of flow.

The camshafts are to be replaced as are the rollers. The rockers will be reused. The camshafts are changed due to material type not profile. Extra fuel delivery is due to injector design not camshaft profile.

The single water pump configuration of the 645E will be replaced with a dual set-up.

The main lube oil and piston cooling pump and the scavenging lube oil pump will be replaced with higher output versions.

A new governor is required with the turbocharged configuration. This has an added air box pressure connection to calibrate for fluctuations in air box pressures

Intake Air

The central air compartment is extensively modified. This is due to an enlarged inertial filter, new engine air filters and the new equipment blower.

The increased intake air demand is being met by increasing the size of the inertial filter on the assistants side of the loco.

The structure on the assistants side of the air room will be modified to account for the new inertial filter and the new equipment blower.

The duct from the inertial filter to the dust bin blower will be replaced with a flexible pipe. The drain for inertial filters will also need relocation.

A new engine air filter assembly will be incorporated to increase the air flow to the engine. The filters will be flexible bag types with a larger surface area. These will be easier to install, even with the reduced installation space.

The equipment blower / auxiliary alternator has been renewed and relocated towards the A side of the loco, as the turbocharger is centrally mounted on the rear end of the engine. The drive is now taken from the right bank camshaft pulley (left side of the loco).
As the auxiliary generator and the traction motor blower are mounted on the same shaft this means the equipment blower will be projecting outside the hood profile. Subsequently a blister to cover the equipment blower will need to be fabricated.

Access down the A/Left side of the loco will be restricted due to the equipment blower cover.

The equipment blower supplies cooling air for the traction motors and the alternator. The alternator air also pressurises the engine compartment and supplies air for the compressor. Due to the relocation of the blower the traction motor air duct and alternator air box will need to be modified.

The traction motor air duct will receive air from the equipment blower from above rather than the side as before, so a 2 metre long section of the duct on the running board, will need replacing.

Modification of the alternator air box will be required so the air intake duct from the equipment blower lines up.

The bulk head between the clean air room and the engine room will also need modification due to the changes.

Cooling

The cooling requirements will be met by a new auxiliary equipment rack, radiator and the new engine ancillaries.

The auxiliary rack houses the increased capacity water header tank, larger oil filter tank, and larger oil cooler.

The new radiator is increased in length and has 8 rows. It will be mounted in the same space but this will be extended forward due to it's extra length. The radiator roof section will be removed to get rid of old brackets and extend the opening. This will include the removal of the loco horn mounted on the long hood, as it is no longer required. The new radiators will need to be shimmed into place.

The original fan will be retained, but the front bulkhead will be modified, so as to avail the radiator of good air flow. This bulkhead will also need an extra hole for the extra radiator pipe. A new section will need to be shaped to seal the new opening. This should be bolted rather than welded for future maintenance of the pipes.

The new radiator grill is the same profile but is 1 3/4 " taller to cover the extra height of the new opening.
The radiator will be off the mechanically bonded type, with fabricated header tanks. The core is of a standard GM type.
Radiator pressure will be 12 psi but may be taken to 20 psi if cooling becomes a problem.

The water system will be changed to a two pipe system running from the two new water pumps on the engine. The radiator, water tanks, screens and pipes etc are all removed. Only the drain pipe remains.
All gaskets, flanges and Marmin couplings will be supplied by GM.

Low water pressure protection incorporates a measure of pressure differential across the pump to get a more accurate result and thus rule out cavitation. The present water pressure check is single take off type.

Bodywork

The external bodywork changes are due to inertial filter changes, radiator modifications, exhaust-turbo hood clearance and equipment blower blister. Also new roof access steps will need to be relocated as their previous location is now taken up by traction motor blower blister.

A new raised centre section on the engine hood will be constructed to clear the turbo and exhaust. This incorporates a rain gutter around exhaust opening.
The exhaust bulge has cut outs to avoid the hinges of the hatches. The rear section of exhaust bulge is welded to roof.
The roof above turbo is to have centre structure removed and a new removable cover constructed.

Electrics

The High voltage cabinet will have 3 modules changed. These are standard items as per the GT22 type GM locos.

These modules will have the effect of boosting the volts-amps control curve for the alternator, to reflect the extra power produced by the engine. The changes are :

Module	Original	New
Throttle Response Reference	TH13	TH14
Feedback	FP15	FP21
Field Shunt	FS14	FS21

The Engine control panel will have an added light for the turbo. Originally this was to show there was power available to the turbo prime/post oil pump. After suggestion by Joe Ruitenbeek it was decided to make this indicate that there is oil pressure to the turbo, which is safer.
An oil pressure switch will be added to the turbo oil supply for this.

The turbo low pressure relay is added to the main control panel.

The turbo soakback pump circuit breaker is mounted on the back of the cabinet door and is fitted with a guard to prevent accidental opening of the circuit and pump stoppage.
The circuit breaker dimensions are yet to be supplied.

The turbo soakback pump is wired across the battery and not the knife switch. This means the pump will work even when the battery switch is open.

A Turbo lube pressure switch is added with a protective shield.

The auxiliary alternator provides 3-phase power which is rectified to excite the main alternator/generator and also to provide 3-phase power for the radiator cooling fan.
The AC cabinet for the fan connection is presently mounted on the existing auxiliary rack in the engine bay. This cabinet is remounted on the new equipment rack via an adapter.

An extra connection board for DJ connections at the rear of the loco is to be removed. The DJ connection is an interface between the 74V of the DF and the 110V of the DJs'.
It is assumed at this stage that the DJ locos are no longer part of the New Zealand Rail Fleet so this connection is not required.

Ballasting

Also as part of the rebuild, extra ballasting will be added to make the end to end weigh equal. This will comprise 50mm plates to a weigh of 600 Kg mounted in the short hood.

Air Compressor

The compressor has been moved back 6¼" to clear the new equipment rack . Also the rear wall of the engine compartment is being modified for the extended radiator, sandwiching it between these two items. Thus the air filter intake for the compressor is being modified with the filters pointing upwards and forwards for clearance.

A new shield for the compressor drive shaft will also need to be manufactured.

Auxiliary Rack

The auxiliary rack is to replaced with a new item incorporating water header tank, oil cooler, oil filter, fuel pump, fuel filter, water temperature manifold.

Low Water Pressure Switch (LWP) from the old water reservoir will be relocated on the new rack.

The thermometer wells from the old pipe work will need to be duplicated on the new pipe work.

Epilogue

The last DF was converted to DFT specification in January 1997. The locomotives will all come back to the workshops to have some extra work carried out over the next year.

- Extra fuel tank added
- Modified drivers window for improved visibility
- Soakback / prelube pump modification
- Spin-on turbocharger lube filter modification
  Oil cooler bypass valve replacement. If required

Appendix B: Soft Mount Alternatives

This section is acknowledged as an excerpt from Venezia [53] on the discussion of soft mount alternative for isolation.

...soft-mounting the cab presents an effective approach to reducing interior noise and vibrations. The mounts effectively work as choking points for the vibration energy. They prevent the energy from reaching to the cab structure and, therefore, can significantly reduce vibrations, as demonstrated in many vehicle applications. The common methods of resiliently mounting a structure include the use of:

- *Elastomeric mounts*
- *Air springs*
- *Fluid mounts*
- *Active mounts*

Each of these elements will be discussed in more detail next.

Elastomeric Mounts

Elastomeric mounts have been used extensively in reducing noise, vibration, and harshness (NVH) since the 1930's. An elastomeric mount is typically a combination of bounded elastomer and metal. The elastomer is generally attached to the metal through a high-pressure and high-temperature bonding process.

The elastomer, as well as the metal and the bounding agent that are used in elastomeric mounts, must be able to withstand the environmental elements, such as temperature variations, cleaning agents, grease, oil, and other contaminants that are often present in vehicles. Furthermore, the mount must be able to support the static weight of the mounted component (i.e., spring body) and also provide dynamic isolation.

The design of elastomeric mounts often involves choosing a mount configuration that satisfies the envelope, environmental, and life requirements. The mount must also provide the proper compromise between the isolation and static load-carrying capability. To support the static weight of the suspended component, the mount must be made sufficiently stiff. For dynamic isolation, however, the mount must be designed as soft as possible. Therefore, mount designers must determine the dynamic input and its frequency composition so that the resonance of the suspended body sitting on the isolation is at a sufficiently low frequency.

This low frequency allows the isolation system to attenuate response to forcing functions that are 1.414 times this mounted resonance frequency. Once they have determined the input frequency range, mount designers select the mount stiffness such that it is rigid enough to carry the load, and yet soft enough to offer isolation in the frequency range of the input.

Air Springs

Air springs were developed during the 1930's at about the same time as the elastomeric pad. The air spring is a carefully designed rubber bellows that contains a column of compressed gas. The air spring elasticity is generated from the gas compression. The rubber is not designed to support load or provide force; this is accomplished by using pressure as the force generation medium. The ability of the spring to support a load depends on the effective area, which is found by dividing the load supported by the gas pressure at a given position. Spring rates may be varied by changing the effective area and/or pressure.

As compared to elastomeric mounts, air springs have a relatively linear spring rate and are able to provide much larger stroke (sometimes by as much as 100 times more stroke). Further, they can be designed to weight less than elastomeric mounts. Air springs, however, have a lower weight-carrying capability than elastomeric mounts, and their spring rate can change with temperature variation. Furthermore, air springs offer no shear

stiffness, unlike elastomeric mounts that can be designed with fairly large shear stiffness.

Fluid Mounts

A fluid mount is simply an elastomeric mount with internal cavities that are filled with fluid. The fluid is able to travel between the cavities (at least two) through a track that can be internal or external to the mount. In the case of a fluid mount with an internal fluid track, the mount is indistinguishable from a regular elastomeric mount.

The mount displacement passes the fluid from one cavity to another through the fluid (inertia) track. This phenomenon causes the rubber surrounding the fluid cavity to bulge. The combination of the fluid inertia and bulge stiffness of the rubber creates a tuned absorber effect that acts in parallel to the elastomer. The tuned absorber effect is amplified by the leverage ratio between the piston area and the area of the inertia track, i.e.,

$$\text{Leverage} = \frac{A_{piston}}{A_{track\ inertia}}$$

The piston area is defined as the effective cross-section of the cavity that the fluid presses against. The tuned absorber acting in parallel to the elastomeric mount creates a dynamic "notch" that causes a lower dynamic stiffness and, therefore, dynamic isolation. The notch is placed within the operating range for the mount. This enables fluid mounts to be statically stiffer, for supporting load, and dynamically softer, for better isolation.

Active Mounts

As was mentioned…, active mounts have been evaluated in several past studies as more advanced alternatives to passive mounts for vibration reduction. Active mounts commonly augment the isolation effect of the passive mounts through the use of actuation elements. The actuation

elements that have commonly been studied range from simple electromechanical actuators to piezoelectric elements to controllable fluids such as electro-rheological and magneto-rheological fluids.

In each case, the actuation mechanism serves two purposes:

1. *it reduces the dynamic stiffness of the mount in the operating range, and*
2. *in most cases, it provides the ability to adapt to varying operating conditions, therefore optimizing the mount effectiveness under all conditions.*

The dynamic stiffness of the active mount has been lowered as compared to passive elastomeric and fluid mounts. Further, the notch frequency for the active mount can be adjusted to accommodate varying operating conditions.

It must be noted, however, that the application of active mounts involves costs and hardware complexities that do not suit many transportation applications. Here, this option was not consider because it falls outside the requirements by the railroad industry for a simple and rugged solution. Indeed, for this study, the simplest solution was chosen, namely elastomeric mounts.

Soft Mounts

To soft mount the cab, it must be completely disconnected from the sill structure and mounted on three or more resilient elements. The resilient elements that can be used for this purpose include:

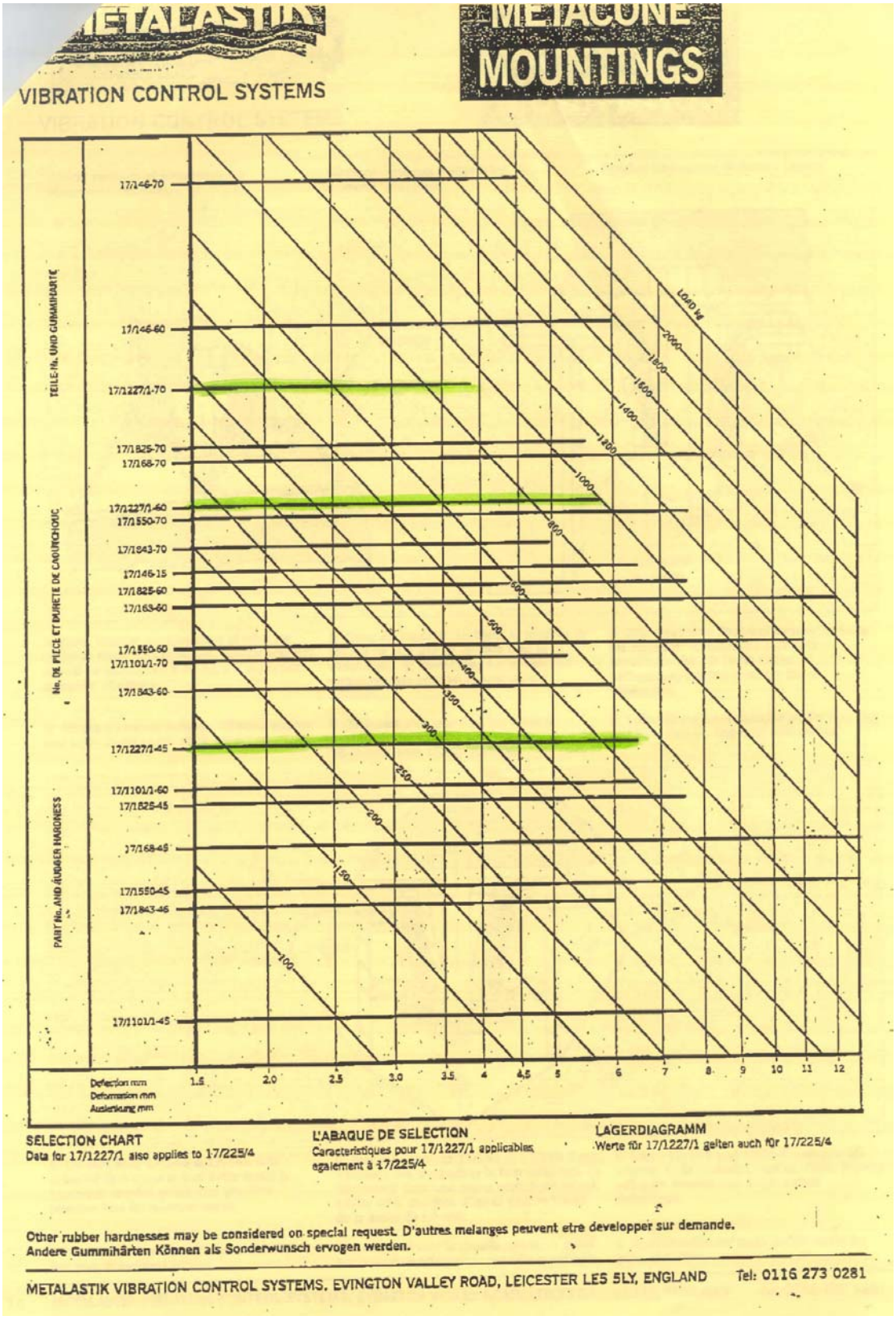
- *rubber mounts*
- *metal springs*
- *air bags*

The resilient elements must be capable of supporting the weight of the cab structure and providing isolation in the dynamic range of isolation. After considering all possibilities, elastomeric mounts were selected for this study. This selection was based on size, installation configuration, shear stability of the cab, and parts availability.

Elastomeric mounts proved to be the most compact, as compared to air bags and coil springs. Although it is possible to design air bags and coil springs that can provide the proper static support and dynamics isolation, they have a larger configuration than elastomeric mounts. This larger size presents problems in installation both for test purposes and actual manufacturing implementation. The installation problems include finding the physical space to accommodate the mounts, which raises the cab too high and, therefore, exceeds the height requirements of the locomotive.

Another problem with air bags and coil springs is shear stability. The resilient element is required to provide isolation in the vertical direction and yet have sufficient stability in the longitudinal and lateral directions (i.e., shear directions of the mount) so that the cab does not experience large displacement when it is subjected to lateral and longitudinal forces. Such forces can occur in coupling and uncoupling the locomotives to the train, or in curving. Air bags and coil springs have a minimal amount of shear stiffness and, therefore, are far less stable than elastomeric mounts in these directions. For elastomeric mounts, it is possible to adjust the shear stiffness properties of the mount through proper design of the mount dimensions, elastomeric compound, and shims.

Appendix C: Tranz Rails' DFT Mounts.



Appendix D: MATLAB Programs

Program Sim2_output.m

```
%for one mass 6 degree of freedom problem
%By Quinton Rowson 3/11/1999
%altered for the transient response of the system on the 2/12/1999
%addpath f:\SAVED\matlab\simulation\sec_sim
%sim2_output

disp('This is the forth simulation version and IS the latest been
used')
clc
clear all
close all
format short g

[m,k1,k2,k3,c1,c2,c3,l1,l2,l3,l4,d1,d2,d3,d4,e1,e2,e3,e4,Ixx,Iyy,Izz,Iy
x,Izx,Izy,Ixy,Ixz,Iyz]=input_variables

%[A,A_nd]=
calc_A_matrix(m,k1,k2,k3,c1,c2,c3,l1,l2,l3,l4,l5,l6,l7,l8,e1,e2,e3,e4,I
x,Iy,Iz);
[AA,K,M,C]=newmass_matrix_6dof(m,k1,k2,k3,c1,c2,c3,l1,l2,d2,d3,l4,l3,d1
,d4,e1,e2,e3,e4,Ixx,Iyy,Izz,Iyx,Izx,Izy,Ixy,Ixz,Iyz);

%[mod,valu]=eig(K,M);
%the matrix AA is obsolete
[r,c]=size(M);
star_M(1:r,1:c)=zeros(size(M));
star_M(1:r,c+1:2*c)=M;
star_M(r+1:2*r,1:c)=M;
star_M(r+1:2*r,c+1:2*c)=C;
star_M%this is the new M matrix with the mass matrix and damping matrix
in it
star_K(1:r,1:c)=-M;
star_K(1:r,c+1:2*c)=zeros(size(M));
star_K(r+1:2*r,1:c)=zeros(size(M));
star_K(r+1:2*r,c+1:2*c)=K;
star_K%this is the new K matrix with the mass matrix and stiffness
matrix in it
[mod,valu]=eig(star_K,star_M);%this give me the eigen-vectors(mod) and
the eigen-values(valu)
%check the accuracy of the eigen-values and eigen-vectors
%to check that the results star_K*mod=star_M*mod*valu and the results
were OK
diag_m_star=mod'*star_M*mod;%this is the unordered unfiltered
diagonalised new M matrix
diag_k_star=mod'*star_K*mod;%this is the unordered unfiltered
diagonalised new K matrix

k=find(abs(diag_k_star)>10^-4);%this starts filter to 4dp
```

```
j=find(abs(diag_m_star)>10^-4);

diag_m_star_prim=zeros(12);
diag_k_star_prim=zeros(12);

diag_m_star_prim(k)=diag_m_star(k);%this if the finished product of the
filtering to 4dp
diag_k_star_prim(j)=diag_k_star(j);

m_permitation=zeros(size(diag_k_star_prim));%This design the size of
the permitation matrix needed to be the same size as the M matrix
m_permitation(find(diag_m_star_prim))=1;% this find all non-zero
elements in a matrix and coverts them all to ones
diag2prime_mstar=m_permitation'*diag_m_star_prim%this is the
diagonalised, ordered, filtered, permuted M matrix

k_permitation=zeros(size(diag_k_star_prim));
k_permitation(find(diag_k_star_prim))=1;
diag2prime_kstar=k_permitation'*diag_k_star_prim%this is the
diagonalised, ordered, filtered, permuted M matrix

%check for the correctness of results
%diag2prime_kstar=valu*diag2prime_mstar
%this was true
%check for the correctness of the mod for the velocity components and
the displacement components
%for i=1:12
%   newmod(:,i)=valu(i,i).*mod(7:12,i)
%   mod(1:6,i)
%end
%this was OK as well as mod(1:6,:)=newmod(:,:)
%is OK

%check for correctness bub=abs(bub)*exp(i*angle(bub)) and this was
correct

alpha=-diag(valu);
frequencies_Hz=imag(alpha)./(2*pi);
mu=real(alpha./((abs(alpha)).^2))
omega=imag(alpha./((abs(alpha)).^2))
theta_M=angle(diag2prime_mstar)
mod_M=abs(diag2prime_mstar)

phi=mod(7:12,:);
theta_phi=angle(phi);
mod_phi=abs(phi);
pick=sign(frequencies_Hz);
i=find(pick==1);
mod_phi=mod_phi(:,i)
theta_phi=theta_phi(:,i)*180/pi
pick=sign(frequencies_Hz);
i=find(pick==1);
frequencies_Hz=frequencies_Hz(i)
```

D2

Program Input_variables

```
Function
[m,k1,k2,k3,c1,c2,c3,l1,l2,l3,l4,d1,d2,d3,d4,e1,e2,e3,e4,Ixx,Iyy,Izz,Iy
x,Izx,Izy,Ixy,Ixz,Iyz]=input_variables

m=3294%(kg) mass of the locomotive cab (this mass has been conformed
30/6/2001)
disp('(kg) mass of the locomotive cab')
k1=1968000%(N/m) axial  dynamic stiffness of mounts, representing the
vertical direction of model
disp('(N/m) axial  dynamic stiffness of mounts, representing the
vertical direction of model')
k2=6891000%(N/m) radial dynamic stiffness of mounts, representing the
lateral direction of model
disp('(N/m) radial dynamic stiffness of mounts, representing the
lateral direction of model')
k3=6891000%(N/m) also radial dynamic stiffness of mounts, representing
the longitudinal direction of model
disp('(N/m) also radial dynamic stiffness of mounts, representing the
longitudinal direction of model')
%************************************************************
%NOTE: That the stuff in between the stars is
%  only for a single degree of freedom system
omega_n(1)=sqrt(k1*4/m);%natural frequency of mount in axial dir
(rad/s)
omega_n(2)=sqrt(k2*4/m);%natural frequency of mount in radial dir
(rad/s)
omega_n(3)=sqrt(k3*4/m);%natural frequency of mount in radial dir
(rad/s)

c_cr(1)=2*sqrt(k1*m/4);%the critical damping for the axial direction of
the mount
c_cr(2)=2*sqrt(k2*m/4);%the critical damping for the radial direction
of the mount
c_cr(3)=2*sqrt(k3*m/4);%the critical damping for the radial direction
of the mount
%w=15;%the frequecny of the cab due to vibration (rad/s)
Q=13;%dynamic magnifier
[c1,c2,c3]=damping_calc(Q,c_cr);
c1
disp('(s.N/m) axial damping of mounts, representing the vertical
direction of model')
c2
disp('(s.N/m) radial damping of mounts, representing the lateral
direction of model')
c3
disp('(s.N/m) radial damping of mounts, representing the longitudinal
direction of model')
%c1=64412%this is a damping ration of 0.8%c1=0
%c2=120528%this is a damping ration of 0.8%c2=0
%c3=120528%this is a damping ration of 0.8%c3=0
%l1=1.272%(m)
%l2=1.625%(m)
%d2=1.132%(m)
```

```
%d3=0.694%(m)
%l4=0.4375%(m)
%l3=0.7825%(m)
%d1=1.132%(m)
%d4=0.694%(m)
%e1=0.842%(m)
%disp('(m) vertical distance for the centre of mass to mount1')
%e2=0.842%(m)
%disp('(m) vertical distance for the centre of mass to mount2')
%e3=0.842%(m)
%disp('(m) vertical distance for the centre of mass to mount3')
%e4=0.842%(m)
%disp('(m) vertical distance for the centre of mass to mount4')

%this was for DFT7145
l1=1.2723%(m)
l2=1.62526%(m)
d2=0.69183%(m)
d3=1.1354%(m)
l4=0.64354%(m)
l3=0.99646%(m)
d1=0.69183%(m)
d4=1.1354%(m)
e1=0.8%(m)
disp('(m) vertical distance for the centre of mass to mount1')
e2=0.8%(m)
disp('(m) vertical distance for the centre of mass to mount2')
e3=0.8%(m)
disp('(m) vertical distance for the centre of mass to mount3')
e4=0.8%(m)
disp('(m) vertical distance for the centre of mass to mount4')
Ixx=3011%(kg/m^2) PITCH of the locomotive cab
disp('(kg/m^2) PITCH of the locomotive cab')
Iyy=2106%(kg/m^2) YAW of the locomotive cab
disp('(kg/m^2) YAW of the locomotive cab')
Izz=3069%(kg/m^2) ROLL of the locomotive cab
disp('(kg/m^2) ROLL of the locomotive cab')
Iyx=-154
Izx=-80
Izy=78
Ixy=-80
Ixz=-154
Iyz=78
```

Program Damping_Calc

```
function [c1,c2,c3]=damping_calc(Q,c_cr)
%DAMPING_CALC  Calculates the the damp of a mount for
%  the for the x,y and z axises provided it is given
%  the mass, stiffness and the DYNAMIC MAGNIFIER of the
%  mounts concerned.
%  The solution that the program looks for is the minimum
%  positive and real solution, as the answer for the damping
%  of the axis concerned.
%
%By Quinton Rowson 20/9/1999
%
%As a part of the requirement for my
%
%Masters of Mechanical Engineering By thesis

for n=1:3
   A=(-4*Q^2)/c_cr(n)^4;
   B=(4*Q^2)/c_cr(n)^2;
   C=-1;
   det(n)=sqrt(B^2-4*A*C);
   c_sqr1(n)=(-B+det(n))/(2*A);
   c_sqr2(n)=(-B-det(n))/(2*A);
   %possible solutions
   no1=sqrt(c_sqr1(n));
   no2=-sqrt(c_sqr1(n));
   no3=sqrt(c_sqr2(n));
   no4=-sqrt(c_sqr2(n));
   [sol]=qadratic_sol_find(no1,no2,no3,no4);
   damp_ratio(n,:)=sol/c_cr(n);
   i(n,:)=find(sol);
end
for n=1:3
   step=1;
   for count=i(n,1):i(n,end)
      if damp_ratio(n,count)>1
         warning('Damping ratio greater than 1 has occured of value:')
         disp(damp_ratio(n,count))
         damp_ratio(n,count)=0;
      end
      if damp_ratio(n,count)>0
         d_r(n,step)=damp_ratio(n,count);
         step=step+1;
      end
   end
end
c1=min(d_r(1,:));
c2=min(d_r(2,:));
c3=min(d_r(3,:));
%c1=max(d_r(1,:));
%c2=max(d_r(2,:));
%c3=max(d_r(3,:));
```

```
c1=c1*c_cr(1);
c2=c2*c_cr(2);
c3=c3*c_cr(3);
```

Program Newmass_matrix_6dof

```
function
[AA,K,M,C]=newmass_matrix_6dof(m,k1,k2,k3,c1,c2,c3,l1,l2,d2,d3,l4,l3,d1
,d4,e1,e2,e3,e4,Ixx,Iyy,Izz,Iyx,Izx,Izy,Ixy,Ixz,Iyz);
%for one mass 6 degree of freedom problem
%By Quinton Rowson 24/5/1999
%mass_matrix_6dof
%
%MASS_MATRIX_6DOF This function sets up the format
%  for the mass matrix of the system, stiffness
%  matrix and the damping matrix. It then uses this
%  to calculate the A matrix used in the state space
%  form called AA.
%
%By Quinton Rowson 21/10/1999
%
%As a part of the requirement for my
%
%Masters of Mechanical Engineering By thesis

%  y  z  x    ry   rz   rx
M=[m, 0, 0,   0,   0,   0;%y
   0, m, 0,   0,   0,   0;%z
   0, 0, m,   0,   0,   0;%x
   0, 0, 0, Ixx, Ixy, Ixz;%ry
   0, 0, 0, Iyx, Iyy, Iyz;%rz
   0, 0, 0, Izx, Izy, Izz];%rx

%damping matrix
%                 y                  z                  x
ry                                              rz
rx
C=[            4*c2,                 0,                 0,
0,                                 c2*(d1+d2-d3-d4),
c2*(e1+e2+e3+e4);%y
                  0,             4*c1,                 0,
c1*(-d1-d2+d3+d4),                                                  0,
c1*(-l1+l2+l3-l4);%z
                  0,                0,              4*c3,
c3*(-e1-e2-e3-e4),                                  c3*(l1-l2-l3+l4),
0;%x
                  0, c1*(-d1-d2+d3+d4), c3*(-e1-e2-e3-e4),
c3*(e1^2+e2^2+e3^2+e4^2)+c1*(d1^2+d2^2+d3^2+d4^2),
c3*(-e1*l1+e2*l2+e3*l3-e4*l4),                      c1*(l1*d1-
l2*d2+d3*l3-l4*d4);%ry
   c2*(d1+d2-d3-d4),                 0,  c3*(l1-l2-l3+l4),
c3*(-e1*l1+e2*l2+e3*l3-e4*l4),
c2*(d1^2+d2^2+d3^2+d4^2)+c3*(l1^2+l2^2+l3^2+l4^2),
c2*(e1*d1+e2*d2-e3*d3-e4*d4);%rz
   c2*(e1+e2+e3+e4), c1*(-l1+l2+l3-l4),               0,
c1*(l1*d1-l2*d2+d3*l3-l4*d4),                     c2*(e1*d1+e2*d2-
e3*d3-e4*d4),c1*(l1^2+l2^2+l3^2+l4^2)+c2*(e1^2+e2^2+e3^2+e4^2)];%rx


%stiffness matrix
```

```
%                 y                   z                   x
ry                                                  rz
rx
K=[            4*k2,                  0,                  0,
0,                                   k2*(d1+d2-d3-d4),
k2*(e1+e2+e3+e4);%y
                  0,              4*k1,                  0,
k1*(-d1-d2+d3+d4),                                                  0,
k1*(-l1+l2+l3-l4);%z
                  0,                 0,              4*k3,
k3*(-e1-e2-e3-e4),                                  k3*(l1-l2-l3+l4),
0;%x
                  0, k1*(-d1-d2+d3+d4), k3*(-e1-e2-e3-e4),
k3*(e1^2+e2^2+e3^2+e4^2)+k1*(d1^2+d2^2+d3^2+d4^2),
k3*(-e1*l1+e2*l2+e3*l3-e4*l4),                      k1*(l1*d1-
l2*d2+d3*l3-l4*d4);%ry
   k2*(d1+d2-d3-d4),                 0,  k3*(l1-l2-l3+l4),
k3*(-e1*l1+e2*l2+e3*l3-e4*l4),
k2*(d1^2+d2^2+d3^2+d4^2)+k3*(l1^2+l2^2+l3^2+l4^2),
k2*(e1*d1+e2*d2-e3*d3-e4*d4);%rz
   k2*(e1+e2+e3+e4), k1*(-l1+l2+l3-l4),                0,
k1*(l1*d1-l2*d2+d3*l3-l4*d4),                      k2*(e1*d1+e2*d2-
e3*d3-e4*d4), k1*(l1^2+l2^2+l3^2+l4^2)+k2*(e1^2+e2^2+e3^2+e4^2)];%rx

%forcing function matrix
%     y     z     x     ry      rz     rx
%F=[  0,   15,    0,     0,      0,     0];

A1_1=zeros(size(M));
A1_2=eye(size(M));
M_inv=inv(M);
A2_1=-M_inv*K;
A2_2=-M_inv*C;

[r,c]=size(M);
AA(1:r,1:c)=A1_1(:,:);
AA(1:r,c+1:2*c)=A1_2(:,:);
AA(r+1:2*r,1:c)=A2_1(:,:);
AA(r+1:2*r,c+1:2*c)=A2_2(:,:);

AA;
```

Program RR26_6_00

```
function RR26_6_00()
%RR26_6_00  This is to calculate that natural frequencies
%using rayleigh ritz of the test bar for ilanko to then
%compare the tne measure experimental result of frequency
%with the use of my accelerometers, data aquisition system,
%FPGA. The results of this test were very encouraging, as
%the experimental results were with in 2 decimal places of
%theoretical results calculated here.

%addpath f:\saved\matlab\fulldata_aqu

L=.825%m
E=207*10^9%Pa
b=.025%m
d=.012%m
I=(b*d^3)/12%m^4
density=7850%kg/m^3
m=b*d*density%mass/unit length
mo=.240%kg
io=80*10^-3%kg/m^2

ilanko_I=io/(m*L^3)
ilanko_m=mo/(m*L)

m11=8*E*I/L^3-2*mo-2*io/(L^2),          k11=(2/5)*m*L;
m22=24*E*I/L^3-2*mo-2*io/(L^2),         k22=(2/7)*m*L;
m33=72*E*I/L^3-2*mo-2*io/(L^2),         k33=(2/9)*m*L;
m44=(400/7)*E*I/L^3-2*mo-2*io/(L^2),    k44=(2/11)*m*L;
m55=200*E*I/L^3-2*mo-2*io/(L^2),     k55=(2/13)*m*L;

m12=12*E*I/L^3-mo-io/(L^2),             k12=m*L/3;
m13=16*E*I/L^3-mo-io/(L^2),             k13=2*m*L/7;
m14=(1/4)*E*I/L^3-mo-io/(L^2),          k14=m*L/4;
m15=24*E*I/L^3-mo-io/(L^2),             k15=2*m*L/9;

m23=36*E*I/L^3-mo-io/(L^2),             k23=m*L/4;
m24=48*E*I/L^3-mo-io/(L^2),             k24=2*m*L/9;
m25=60*E*I/L^3-mo-io/(L^2),             k25=m*L/5;

m34=80*E*I/L^3-mo-io/(L^2),             k34=m*L/5;
m35=(720/7)*E*I/L^3-mo-io/(L^2),     k35=2*m*L/11;

m45=150*E*I/L^3-mo-io/(L^2),            k45=m*L/6;

m21=m12;
m31=m13;
m41=m14;
m51=m15;

m32=m23;
m42=m24;
m52=m25;
```

```
m43=m34;
m53=m35;

m54=m45;

k21=k12;
k31=k13;
k41=k14;
k51=k15;

k32=k23;
k42=k24;
k52=k25;

k43=k34;
k53=k35;

k54=k45;


M=[m11, m12, m13, m14, m15;
   m21, m22, m23, m24, m25;
   m31, m32, m33, m34, m35;
   m41, m42, m43, m44, m45;
   m51, m52, m53, m54, m55]./1000000000

K=[k11, k12, k13, k14, k15;
   k21, k22, k23, k24, k25;
   k31, k32, k33, k34, k35;
   k41, k42, k43, k44, k45;
   k51, k52, k53, k54, k55]

omega=sqrt(eig(K,M))/(2*pi)*100

lamda1=1.5642578
lamda2=2.7912109
lamda3=5.3429688
lamda4=8.3349609

freq1=sqrt((E*I/m)*(lamda1/L)^4)/(2*pi)
freq2=sqrt((E*I/m)*(lamda2/L)^4)/(2*pi)
freq3=sqrt((E*I/m)*(lamda3/L)^4)/(2*pi)
freq4=sqrt((E*I/m)*(lamda4/L)^4)/(2*pi)
```

Appendix E: Specifications of the Accelerometer

ANALOG DEVICES

Low Cost ±2 *g*/±10 *g* Dual Axis *i*MEMS® Accelerometers with Digital Output

ADXL202/ADXL210

FEATURES

2-Axis Acceleration Sensor on a Single IC Chip
Measures Static Acceleration as Well as Dynamic Acceleration
Duty Cycle Output with User Adjustable Period
Low Power <0.6 mA
Faster Response than Electrolytic, Mercury or Thermal Tilt Sensors
Bandwidth Adjustment with a Single Capacitor Per Axis
5 m*g* Resolution at 60 Hz Bandwidth
+3 V to +5.25 V Single Supply Operation
1000 *g* Shock Survival

APPLICATIONS

2-Axis Tilt Sensing
Computer Peripherals
Inertial Navigation
Seismic Monitoring
Vehicle Security Systems
Battery Powered Motion Sensing

GENERAL DESCRIPTION

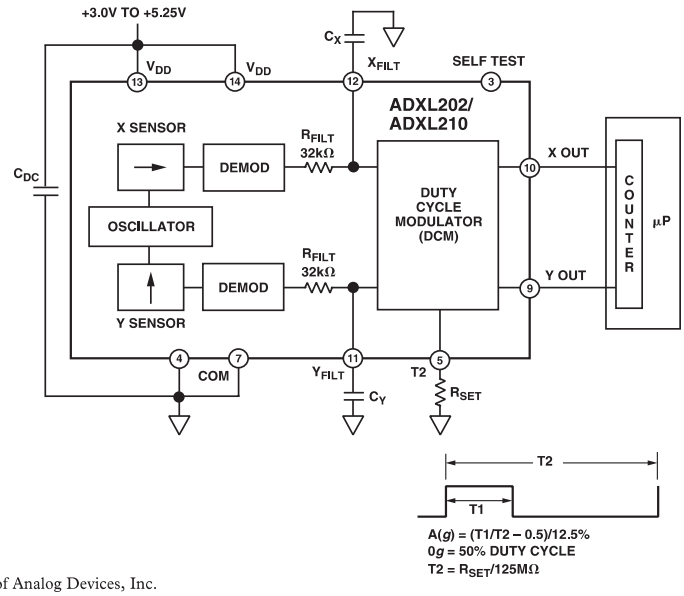
The ADXL202/ADXL210 are low cost, low power, complete 2-axis accelerometers with a measurement range of either ±2 *g*/±10 *g*. The ADXL202/ADXL210 can measure both dynamic acceleration (e.g., vibration) and static acceleration (e.g., gravity).

The outputs are digital signals whose duty cycles (ratio of pulsewidth to period) are proportional to the acceleration in each of the 2 sensitive axes. These outputs may be measured directly with a microprocessor counter, requiring no A/D converter or glue logic. The output period is adjustable from 0.5 ms to 10 ms via a single resistor (R_{SET}). If a voltage output is desired, a voltage output proportional to acceleration is available from the X_{FILT} and Y_{FILT} pins, or may be reconstructed by filtering the duty cycle outputs.

The bandwidth of the ADXL202/ADXL210 may be set from 0.01 Hz to 5 kHz via capacitors C_X and C_Y. The typical noise floor is 500 μg/√Hz allowing signals below 5 m*g* to be resolved for bandwidths below 60 Hz.

The ADXL202/ADXL210 is available in a hermetic 14-lead Surface Mount CERPAK, specified over the 0°C to +70°C commercial or –40°C to +85°C industrial temperature range.

FUNCTIONAL BLOCK DIAGRAM



*i*MEMS is a registered trademark of Analog Devices, Inc.

REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703

ADXL202/ADXL210–SPECIFICATIONS ($T_A = T_{MIN}$ to T_{MAX}, T_A = +25°C for J Grade only, V_{DD} = +5 V, R_{SET} = 125 kΩ, Acceleration = 0 *g*, unless otherwise noted)

Parameter	Conditions	ADXL202/JQC/AQC Min	Typ	Max	ADXL210/JQC/AQC Min	Typ	Max	Units
SENSOR INPUT	Each Axis							
Measurement Range[1]		±1.5	±2		±8	±10		*g*
Nonlinearity	Best Fit Straight Line		0.2			0.2		% of FS
Alignment Error[2]			±1			±1		Degrees
Alignment Error	X Sensor to Y Sensor		±0.01			±0.01		Degrees
Transverse Sensitivity[3]			±2			±2		%
SENSITIVITY	Each Axis							
Duty Cycle per *g*	T1/T2 @ +25°C	10	12.5	15	3.2	4.0	4.8	%/*g*
Sensitivity, Analog Output	At Pins X_{FILT}, Y_{FILT}		312			100		mV/*g*
Temperature Drift[4]	Δ from +25°C		±0.5			±0.5		% Rdg
ZERO *g* BIAS LEVEL	Each Axis							
0 *g* Duty Cycle	T1/T2	25	50	75	42	50	58	%
Initial Offset			±2			±2		*g*
0 *g* Duty Cycle vs. Supply			1.0	4.0		1.0	4.0	%/V
0 *g* Offset vs. Temperature[4]	Δ from +25°C		2.0			2.0		m*g*/°C
NOISE PERFORMANCE								
Noise Density[5]	@ +25°C		500	1000		500	1000	µ*g*/√Hz
FREQUENCY RESPONSE								
3 dB Bandwidth	Duty Cycle Output		500			500		Hz
3 dB Bandwidth	At Pins X_{FILT}, Y_{FILT}		5			5		kHz
Sensor Resonant Frequency			10			14		kHz
FILTER								
R_{FILT} Tolerance	32 kΩ Nominal		±15			±15		%
Minimum Capacitance	At X_{FILT}, Y_{FILT}	1000			1000			pF
SELF TEST								
Duty Cycle Change	Self-Test "0" to "1"		10			10		%
DUTY CYCLE OUTPUT STAGE								
F_{SET}			125 MΩ/R_{SET}			125 MΩ/R_{SET}		
F_{SET} Tolerance	R_{SET} = 125 kΩ	0.7		1.3	0.7		1.3	kHz
Output High Voltage	I = 25 µA	V_S – 200 mV			V_S – 200 mV			mV
Output Low Voltage	I = 25 µA			200			200	mV
T2 Drift vs. Temperature			35			35		ppm/°C
Rise/Fall Time			200			200		ns
POWER SUPPLY								
Operating Voltage Range		3.0		5.25	2.7		5.25	V
Specified Performance		4.75		5.25	4.75		5.25	V
Quiescent Supply Current			0.6	1.0		0.6	1.0	mA
Turn-On Time[6]	To 99%		160 C_{FILT} + 0.3			160 C_{FILT} + 0.3		ms
TEMPERATURE RANGE								
Operating Range	JQC	0		+70	0		+70	°C
Specified Performance	AQC	–40		+85	–40		+85	°C

NOTES
[1]For all combinations of offset and sensitivity variation.
[2]Alignment error is specified as the angle between the true and indicated axis of sensitivity.
[3]Transverse sensitivity is the algebraic sum of the alignment and the inherent sensitivity errors.
[4]Specification refers to the maximum change in parameter from its initial at +25 °C to its worst case value at T_{MIN} to T_{MAX}.
[5]Noise density (µ*g*/√Hz) is the average noise at any frequency in the bandwidth of the part.
[6]C_{FILT} in µF. Addition of filter capacitor will increase turn on time. Please see the Application section on power cycling.

All min and max specifications are guaranteed. Typical specifications are not tested or guaranteed.
Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS*

Acceleration (Any Axis, Unpowered for 0.5 ms) ..... 1000 *g*
Acceleration (Any Axis, Powered for 0.5 ms) ..... 500 *g*
$+V_S$ ..... –0.3 V to +7.0 V
Output Short Circuit Duration
(Any Pin to Common) ..... Indefinite
Operating Temperature .....–55°C to +125°C
Storage Temperature .....–65°C to +150°C

*Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Drops onto hard surfaces can cause shocks of greater than 1000 *g* and exceed the absolute maximum rating of the device. Care should be exercised in handling to avoid damage.

PIN FUNCTION DESCRIPTIONS

Pin	Name	Description
1	NC	No Connect
2	V_{TP}	Test Point, Do Not Connect
3	ST	Self Test
4	COM	Common
5	T2	Connect R_{SET} to Set T2 Period
6	NC	No Connect
7	COM	Common
8	NC	No Connect
9	Y_{OUT}	Y Axis Duty Cycle Output
10	X_{OUT}	X Axis Duty Cycle Output
11	Y_{FILT}	Connect Capacitor for Y Filter
12	X_{FILT}	Connect Capacitor for X Filter
13	V_{DD}	+3 V to +5.25 V, Connect to 14
14	V_{DD}	+3 V to +5.25 V, Connect to 13

PACKAGE CHARACTERISTICS

Package	θ_{JA}	θ_{JC}	Device Weight
14-Lead CERPAK	110°C/W	30°C/W	5 Grams

PIN CONFIGURATION

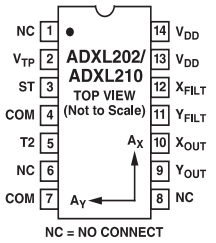


Figure 1 shows the response of the ADXL202 to the Earth's gravitational field. The output values shown are nominal. They are presented to show the user what type of response to expect from each of the output pins due to changes in orientation with respect to the Earth. The ADXL210 reacts similarly with output changes appropriate to its scale.

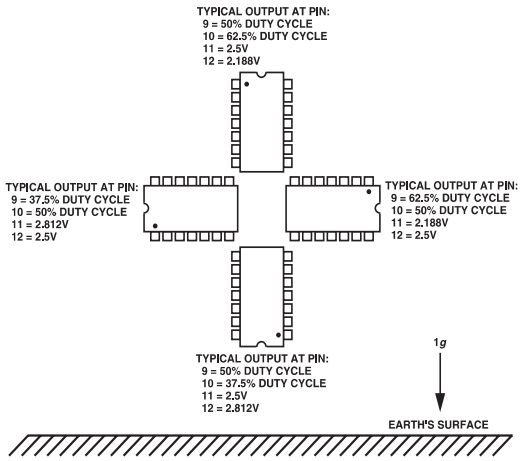


Figure 1. ADXL202/ADXL210 Nominal Response Due to Gravity

ORDERING GUIDE

Model	*g* Range	Temperature Range	Package Description	Package Option
ADXL202JQC	±2	0°C to +70°C	14-Lead CERPAK	QC-14
ADXL202AQC	±2	–40°C to +85°C	14-Lead CERPAK	QC-14
ADXL210JQC	±10	0°C to +70°C	14-Lead CERPAK	QC-14
ADXL210AQC	±10	–40°C to +85°C	14-Lead CERPAK	QC-14

CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the ADXL202/ADXL210 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

TYPICAL CHARACTERISTICS (@ +25°C R_{SET} = 125 kΩ, V_{DD} = +5 V, unless otherwise noted)

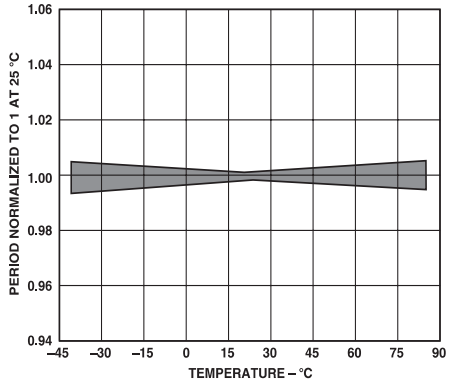


Figure 2. Normalized DCM Period (T2) vs. Temperature

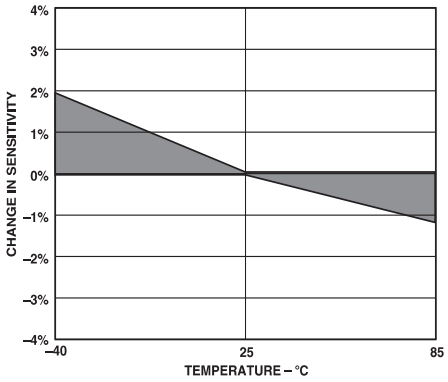


Figure 5. Typical X Axis Sensitivity Drift Due to Temperature

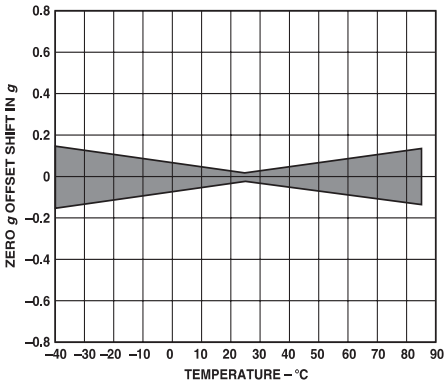


Figure 3. Typical Zero g Offset vs. Temperature

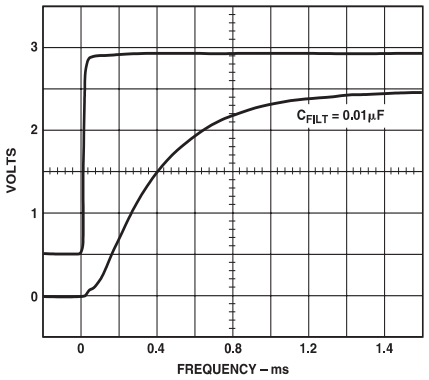


Figure 6. Typical Turn-On Time

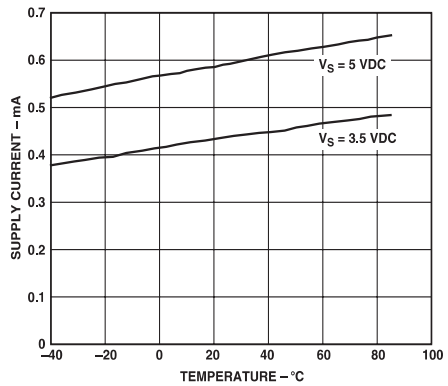


Figure 4. Typical Supply Current vs. Temperature

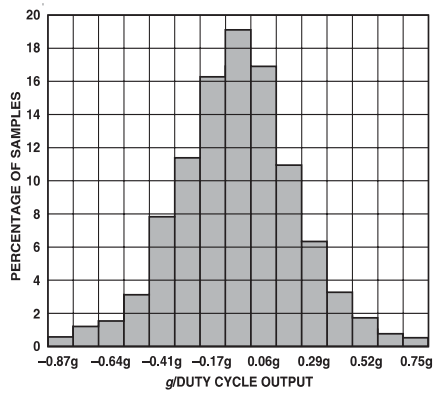


Figure 7. Typical Zero g Distribution at +25°C

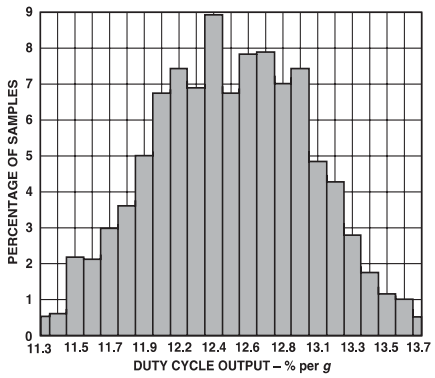


Figure 8. Typical Sensitivity per g at +25°C

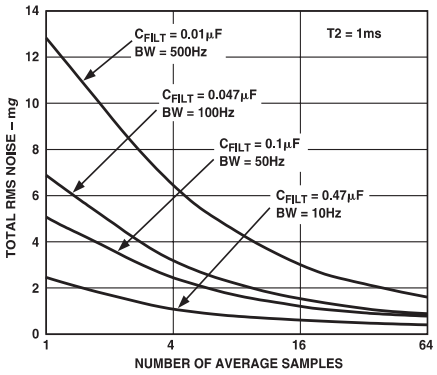


Figure 10. Typical Noise at Digital Outputs

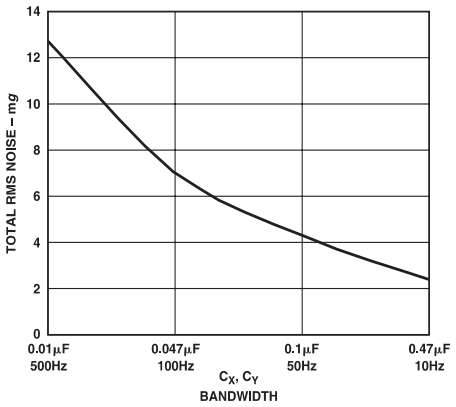


Figure 9. Typical Noise at X_{FILT} Output

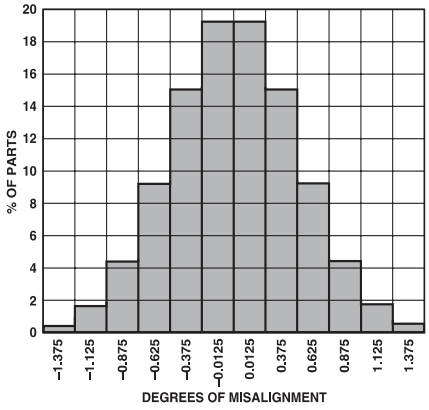


Figure 11. Rotational Die Alignment

DEFINITIONS

T1	Length of the "on" portion of the cycle.
T2	Length of the total cycle.
Duty Cycle	Ratio of the "on" time (T1) of the cycle to the total cycle (T2). Defined as T1/T2 for the ADXL202/ADXL210.
Pulsewidth	Time period of the "on" pulse. Defined as T1 for the ADXL202/ADXL210.

THEORY OF OPERATION

The ADXL202/ADXL210 are complete dual axis acceleration measurement systems on a single monolithic IC. They contain a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open loop acceleration measurement architecture. For each axis, an output circuit converts the analog signal to a duty cycle modulated (DCM) digital signal that can be decoded with a counter/timer port on a microprocessor. The ADXL202/ADXL210 are capable of measuring both positive and negative accelerations to a maximum level of ±2 *g* or ±10 *g*. The accelerometer measures static acceleration forces such as gravity, allowing it to be used as a tilt sensor.

The sensor is a surface micromachined polysilicon structure built on top of the silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and central plates attached to the moving mass. The fixed plates are driven by 180° out of phase square waves. An acceleration will deflect the beam and unbalance the differential capacitor, resulting in an output square wave whose amplitude is proportional to acceleration. Phase sensitive demodulation techniques are then used to rectify the signal and determine the direction of the acceleration.

The output of the demodulator drives a duty cycle modulator (DCM) stage through a 32 kΩ resistor. At this point a pin is available on each channel to allow the user to set the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

After being low-pass filtered, the analog signal is converted to a duty cycle modulated signal by the DCM stage. A single resistor sets the period for a complete cycle (T2), which can be set between 0.5 ms and 10 ms (see Figure 12). A 0 *g* acceleration produces a nominally 50% duty cycle. The acceleration signal can be determined by measuring the length of the T1 and T2 pulses with a counter/timer or with a polling loop using a low cost microcontroller.

An analog output voltage can be obtained either by buffering the signal from the X_{FILT} and Y_{FILT} pin, or by passing the duty cycle signal through an RC filter to reconstruct the dc value.

The ADXL202/ADXL210 will operate with supply voltages as low as 3.0 V or as high as 5.25 V.

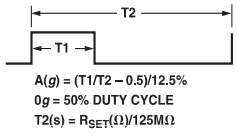


Figure 12. Typical Output Duty Cycle

APPLICATIONS

POWER SUPPLY DECOUPLING

For most applications a single 0.1 µF capacitor, C_{DC}, will adequately decouple the accelerometer from signal and noise on the power supply. However, in some cases, especially where digital devices such as microcontrollers share the same power supply, digital noise on the supply may cause interference on the ADXL202/ADXL210 output. This is often observed as a slowly undulating fluctuation of voltage at X_{FILT} and Y_{FILT}. If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite beads, may be inserted in the ADXL202/ADXL210's supply line.

DESIGN PROCEDURE FOR THE ADXL202/ADXL210

The design procedure for using the ADXL202/ADXL210 with a duty cycle output involves selecting a duty cycle period and a filter capacitor. A proper design will take into account the application requirements for bandwidth, signal resolution and acquisition time, as discussed in the following sections.

V_{DD}

The ADXL202/ADXL210 have two power supply (V_{DD}) Pins: 13 and 14. These two pins should be connected directly together.

COM

The ADXL202/ADXL210 have two commons, Pins 4 and 7. These two pins should be connected directly together and Pin 7 grounded.

V_{TP}

This pin is to be left open; make no connections of any kind to this pin.

Decoupling Capacitor C_{DC}

A 0.1 µF capacitor is recommended from V_{DD} to COM for power supply decoupling.

ST

The ST pin controls the self-test feature. When this pin is set to V_{DD}, an electrostatic force is exerted on the beam of the accelerometer. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output will be 10% at the duty cycle outputs (corresponding to 800 mg). This pin may be left open circuit or connected to common in normal use.

Duty Cycle Decoding

The ADXL202/ADXL210's digital output is a duty cycle modulator. Acceleration is proportional to the ratio T1/T2. The nominal output of the ADXL202 is:

$$0\ g = 50\%\ Duty\ Cycle$$

Scale factor is 12.5% Duty Cycle Change per *g*

The nominal output of the ADXL210 is:

$$0\ g = 50\%\ Duty\ Cycle$$

Scale factor is 4% Duty Cycle Change per *g*

These nominal values are affected by the initial tolerance of the device including zero *g* offset error and sensitivity error.

T2 does not have to be measured for every measurement cycle. It need only be updated to account for changes due to temperature, (a relatively slow process). Since the T2 time period is shared by both X and Y channels, it is necessary only to measure it on one channel of the ADXL202/ADXL210. Decoding algorithms for various microcontrollers have been developed. Consult the appropriate Application Note.

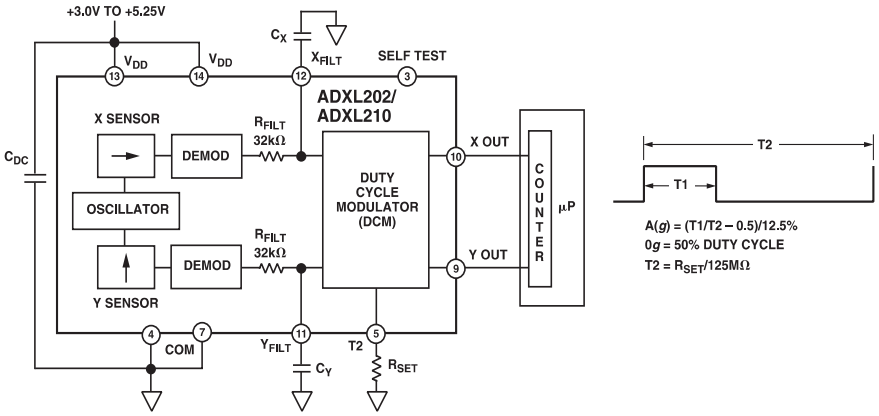


Figure 13. Block Diagram

Setting the Bandwidth Using C_X and C_Y

The ADXL202/ADXL210 have provisions for bandlimiting the X_{FILT} and Y_{FILT} pins. Capacitors must be added at these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwidth is:

$$F_{-3\,dB} = \frac{1}{\left(2\,\pi\,(32\,k\Omega) \times C(x,y)\right)}$$

or, more simply, $F_{-3\,dB} = \frac{5\,\mu F}{C_{(X,Y)}}$

The tolerance of the internal resistor (R_{FILT}), can vary as much as ±25% of its nominal value of 32 kΩ; so the bandwidth will vary accordingly. A minimum capacitance of 1000 pF for $C_{(X, Y)}$ is required in all cases.

Table I. Filter Capacitor Selection, C_X and C_Y

Bandwidth	Capacitor Value
10 Hz	0.47 µF
50 Hz	0.10 µF
100 Hz	0.05 µF
200 Hz	0.027 µF
500 Hz	0.01 µF
5 kHz	0.001 µF

Setting the DCM Period with R_{SET}

The period of the DCM output is set for both channels by a single resistor from R_{SET} to ground. The equation for the period is:

$$T2 = \frac{R_{SET}\,(\Omega)}{125\,M\Omega}$$

A 125 kΩ resistor will set the duty cycle repetition rate to approximately 1 kHz, or 1 ms. The device is designed to operate at duty cycle periods between 0.5 ms and 10 ms.

Table II. Resistor Values to Set T2

T2	R_{SET}
1 ms	125 kΩ
2 ms	250 kΩ
5 ms	625 kΩ
10 ms	1.25 MΩ

Note that the R_{SET} should always be included, even if only an analog output is desired. Use an R_{SET} value between 500 kΩ and 2 MΩ when taking the output from X_{FILT} or Y_{FILT}. The R_{SET} resistor should be place close to the T2 Pin to minimize parasitic capacitance at this node.

Selecting the Right Accelerometer

For most tilt sensing applications the ADXL202 is the most appropriate accelerometer. Its higher sensitivity (12.5%/*g* allows the user to use a lower speed counter for PWM decoding while maintaining high resolution. The ADXL210 should be used in applications where accelerations of greater than ±2 *g* are expected.

MICROCOMPUTER INTERFACES

The ADXL202/ADXL210 were specifically designed to work with low cost microcontrollers. Specific code sets, reference designs, and application notes are available from the factory. This section will outline a general design procedure and discuss the various trade-offs that need to be considered.

The designer should have some idea of the required performance of the system in terms of:

Resolution: the smallest signal change that needs to be detected.

Bandwidth: the highest frequency that needs to be detected.

Acquisition Time: the time that will be available to acquire the signal on each axis.

These requirements will help to determine the accelerometer bandwidth, the speed of the microcontroller clock and the length of the T2 period.

When selecting a microcontroller it is helpful to have a counter timer port available. The microcontroller should have provisions for software calibration. While the ADXL202/ADXL210 are highly accurate accelerometers, they have a wide tolerance for

initial offset. The easiest way to null this offset is with a calibration factor saved on the microcontroller or by a user calibration for zero *g*. In the case where the offset is calibrated during manufacture, there are several options, including external EEPROM and microcontrollers with "one-time programmable" features.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The accelerometer bandwidth selected will determine the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor and improve the resolution of the accelerometer. Resolution is dependent on both the analog filter bandwidth at X_{FILT} and Y_{FILT} and on the speed of the microcontroller counter.

The analog output of the ADXL202/ADXL210 has a typical bandwidth of 5 kHz, much higher than the duty cycle stage is capable of converting. The user must filter the signal at this point to limit aliasing errors. To minimize DCM errors the analog bandwidth should be less than 1/10 the DCM frequency. Analog bandwidth may be increased to up to 1/2 the DCM frequency in many applications. This will result in greater dynamic error generated at the DCM.

The analog bandwidth may be further decreased to reduce noise and improve resolution. The ADXL202/ADXL210 noise has the characteristics of white Gaussian noise that contributes equally at all frequencies and is described in terms of µg per root Hz; i.e., the noise is proportional to the square root of the bandwidth of the accelerometer. It is recommended that the user limit bandwidth to the lowest frequency needed by the application, to maximize the resolution and dynamic range of the accelerometer.

With the single pole roll-off characteristic, the typical noise of the ADXL202/ADXL210 is determined by the following equation:

$$Noise\ (rms) = \left(500\ \mu g/\sqrt{Hz}\right) \times \left(\sqrt{BW \times 1.5}\right)$$

At 100 Hz the noise will be:

$$Noise\ (rms) = \left(500\ \mu g/\sqrt{Hz}\right) \times \left(\sqrt{100 \times (1.5)}\right) = 6.12\ mg$$

Often the peak value of the noise is desired. Peak-to-peak noise can only be estimated by statistical methods. Table III is useful for estimating the probabilities of exceeding various peak values, given the rms value.

Table III. Estimation of Peak-to-Peak Noise

Nominal Peak-to-Peak Value	% of Time that Noise Will Exceed Nominal Peak-to-Peak Value
2.0 × rms	32%
4.0 × rms	4.6%
6.0 × rms	0.27%
8.0 × rms	0.006%

The peak-to-peak noise value will give the best estimate of the uncertainty in a single measurement.

Table IV gives typical noise output of the ADXL202/ADXL210 for various C_X and C_Y values.

Table IV. Filter Capacitor Selection, C_X and C_Y

Bandwidth	C_X, C_Y	rms Noise	Peak-to-Peak Noise Estimate 95% Probability (rms × 4)
10 Hz	0.47 µF	1.9 m*g*	7.6 m*g*
50 Hz	0.10 µF	4.3 m*g*	17.2 m*g*
100 Hz	0.05 µF	6.1 m*g*	24.4 m*g*
200 Hz	0.027 µF	8.7 m*g*	35.8 m*g*
500 Hz	0.01 µF	13.7 m*g*	54.8 m*g*

CHOOSING T2 AND COUNTER FREQUENCY: DESIGN TRADE-OFFS

The noise level is one determinant of accelerometer resolution. The second relates to the measurement resolution of the counter when decoding the duty cycle output.

The ADXL202/ADXL210's duty cycle converter has a resolution of approximately 14 bits; better resolution than the accelerometer itself. The actual resolution of the acceleration signal is, however, limited by the time resolution of the counting devices used to decode the duty cycle. The faster the counter clock, the higher the resolution of the duty cycle and the shorter the T2 period can be for a given resolution. The following table shows some of the trade-offs. It is important to note that this is the resolution due to the microprocessors's counter. It is probable that the accelerometer's noise floor may set the lower limit on the resolution, as discussed in the previous section.

Table V. Trade-Offs Between Microcontroller Counter Rate, T2 Period and Resolution of Duty Cycle Modulator

T2 (ms)	R_{SET} (kΩ)	ADXL202/ ADXL210 Sample Rate	Counter-Clock Rate (MHz)	Counts per T2 Cycle	Counts per *g*	Resolution (m*g*)
1.0	124	1000	2.0	2000	250	4.0
1.0	124	1000	1.0	1000	125	8.0
1.0	124	1000	0.5	500	62.5	16.0
5.0	625	200	2.0	10000	1250	0.8
5.0	625	200	1.0	5000	625	1.6
5.0	625	200	0.5	2500	312.5	3.2
10.0	1250	100	2.0	20000	2500	0.4
10.0	1250	100	1.0	10000	1250	0.8
10.0	1250	100	0.5	5000	625	1.6

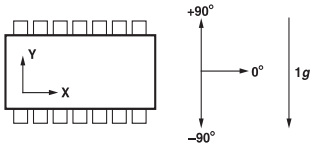
STRATEGIES FOR USING THE DUTY CYCLE OUTPUT WITH MICROCONTROLLERS

Application notes outlining various strategies for using the duty cycle output with low cost microcontrollers are available from the factory.

USING THE ADXL202/ADXL210 AS A DUAL AXIS TILT SENSOR

One of the most popular applications of the ADXL202/ADXL210 is tilt measurement. An accelerometer uses the force of gravity as an input vector to determine orientation of an object in space.

An accelerometer is most sensitive to tilt when its sensitive axis is perpendicular to the force of gravity, i.e., parallel to the earth's surface. At this orientation its sensitivity to changes in tilt is highest. When the accelerometer is oriented on axis to gravity, i.e., near its +1 *g* or –1 *g* reading, the change in output acceleration per degree of tilt is negligible. When the accelerometer is perpendicular to gravity, its output will change nearly 17.5 m*g* per degree of tilt, but at 45° degrees it is changing only at 12.2 m*g* per degree and resolution declines. The following table illustrates the changes in the X and Y axes as the device is tilted ±90° through gravity.



X AXIS ORIENTATION TO HORIZON (°)	X OUTPUT (*g*)	X OUTPUT Δ PER DEGREE OF TILT (m*g*)	Y OUTPUT (*g*)	Y OUTPUT Δ PER DEGREE OF TILT (m*g*)
–90	–1.000	–0.2	0.000	17.5
–75	–0.966	4.4	0.259	16.9
–60	–0.866	8.6	0.500	15.2
–45	–0.707	12.2	0.707	12.4
–30	–0.500	15.0	0.866	8.9
–15	–0.259	16.8	0.966	4.7
0	0.000	17.5	1.000	0.2
15	0.259	16.9	0.966	–4.4
30	0.500	15.2	0.866	–8.6
45	0.707	12.4	0.707	–12.2
60	0.866	8.9	0.500	–15.0
75	0.966	4.7	0.259	–16.8
90	1.000	0.2	0.000	–17.5

Figure 14. How the X and Y Axes Respond to Changes in Tilt

A DUAL AXIS TILT SENSOR: CONVERTING ACCELERATION TO TILT

When the accelerometer is oriented so both its X and Y axes are parallel to the earth's surface it can be used as a two axis tilt sensor with a roll and a pitch axis. Once the output signal from the accelerometer has been converted to an acceleration that varies between –1 *g* and +1 *g*, the output tilt in degrees is calculated as follows:

$$Pitch = ASIN\ (Ax/1\ g)$$
$$Roll = ASIN\ (Ay/1\ g)$$

Be sure to account for overranges. It is possible for the accelerometers to output a signal greater than ±1 *g* due to vibration, shock or other accelerations.

MEASURING 360° OF TILT

It is possible to measure a full 360° of orientation through gravity by using two accelerometers oriented perpendicular to one another (see Figure 15). When one sensor is reading a maximum change in output per degree, the other is at its minimum.

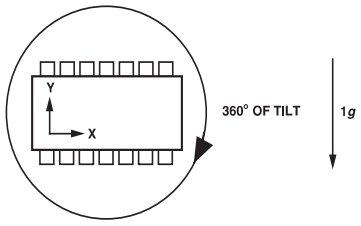


Figure 15. Using a Two-Axis Accelerometer to Measure 360° of Tilt

USING THE ANALOG OUTPUT

The ADXL202/ADXL210 was specifically designed for use with its digital outputs, but has provisions to provide analog outputs as well.

Duty Cycle Filtering

An analog output can be reconstructed by filtering the duty cycle output. This technique requires only passive components. The duty cycle period (T2) should be set to 1 ms. An RC filter with a 3 dB point at least a factor of 10 less than the duty cycle frequency is connected to the duty cycle output. The filter resistor should be no less than 100 kΩ to prevent loading of the output stage. The analog output signal will be ratiometric to the supply voltage. The advantage of this method is an output scale factor of approximately double the analog output. Its disadvantage is that the frequency response will be lower than when using the X_{FILT}, Y_{FILT} output.

X_{FILT}, Y_{FILT} Output

The second method is to use the analog output present at the X_{FILT} and Y_{FILT} pin. Unfortunately, these pins have a 32 kΩ output impedance and are not designed to drive a load directly. An op amp follower may be required to buffer this pin. The advantage of this method is that the full 5 kHz bandwidth of the accelerometer is available to the user. A capacitor still must be added at this point for filtering. The duty cycle converter should be kept running by using R_{SET} <10 MΩ. Note that the accelerometer offset and sensitivity are ratiometric to the supply voltage. The offset and sensitivity are nominally:

0 *g* Offset = $V_{DD}/2$	2.5 V at +5 V
ADXL202 Sensitivity = (60 mV × V_S)/*g*	300 mV/*g* at +5 V, V_{DD}
ADXL210 Sensitivity = (20 mV × V_S)/*g*	100 mV/*g* at +5 V, V_{DD}

USING THE ADXL202/ADXL210 IN VERY LOW POWER APPLICATIONS

An application note outlining low power strategies for the ADXL202/ADXL210 is available. Some key points are presented here. It is possible to reduce the ADXL202/ADXL210's average current from 0.6 mA to less than 20 µA by using the following techniques:

1. Power Cycle the accelerometer.
2. Run the accelerometer at a Lower Voltage, (Down to 3 V).

Power Cycling with an External A/D

Depending on the value of the X_{FILT} capacitor, the ADXL202/ADXL210 is capable of turning on and giving a good reading in 1.6 ms. Most microcontroller based A/Ds can acquire a reading in another 25 µs. Thus it is possible to turn on the ADXL202/ADXL210 and take a reading in <2 ms. If we assume that a 20 Hz sample rate is sufficient, the total current required to take 20 samples is 2 ms × 20 samples/s × 0.6 mA = 24 µA average current. Running the part at 3 V will reduce the supply current from 0.6 mA to 0.4 mA, bringing the average current down to 16 µA.

The A/D should read the analog output of the ADXL202/ADXL210 at the X_{FILT} and Y_{FILT} pins. A buffer amplifier is recommended, and may be required in any case to amplify the analog output to give enough resolution with an 8-bit to 10-bit converter.

Power Cycling When Using the Digital Output

An alternative is to run the microcontroller at a higher clock rate and put it into shutdown between readings, allowing the use of the digital output. In this approach the ADXL202/ADXL210 should be set at its fastest sample rate (T2 = 0.5 ms), with a 500 Hz filter at X_{FILT} and Y_{FILT}. The concept is to acquire a reading as quickly as possible and then shut down the ADXL202/ADXL210 and the microcontroller until the next sample is needed.

In either of the above approaches, the ADXL202/ADXL210 can be turned on and off directly using a digital port pin on the microcontroller to power the accelerometer without additional components. The port should be used to switch the common pin of the accelerometer so the port pin is "pulling down."

CALIBRATING THE ADXL202/ADXL210

The initial value of the offset and scale factor for the ADXL202/ADXL210 will require calibration for applications such as tilt measurement. The ADXL202/ADXL210 architecture has been designed so that these calibrations take place in the software of the microcontroller used to decode the duty cycle signal. Calibration factors can be stored in EEPROM or determined at turn-on and saved in dynamic memory.

For low *g* applications, the force of gravity is the most stable, accurate and convenient acceleration reference available. A reading of the 0 *g* point can be determined by orientating the device parallel to the earth's surface and then reading the output.

A more accurate calibration method is to make a measurements at +1 *g* and –1 *g*. The sensitivity can be determined by the two measurements.

To calibrate, the accelerometer's measurement axis is pointed directly at the earth. The 1 *g* reading is saved and the sensor is turned 180° to measure –1 *g*. Using the two readings, the sensitivity is:

Let A = Accelerometer output with axis oriented to +1 *g*
Let B = Accelerometer output with axis oriented to –1 *g then*:
Sensitivity = [A – *B*]/2 *g*

For example, if the +1 *g* reading (A) is 55% duty cycle and the –1 *g* reading (B) is 32% duty cycle, then:

Sensitivity = [55% – 32%]/2 *g* = 11.5%/*g*

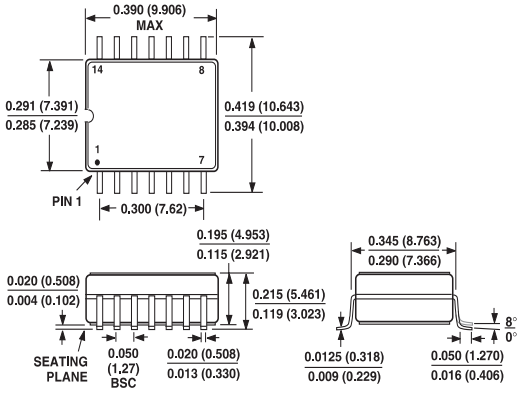
These equations apply whether the output is analog, or duty cycle.

Application notes outlining algorithms for calculating acceleration from duty cycle and automated calibration routines are available from the factory.

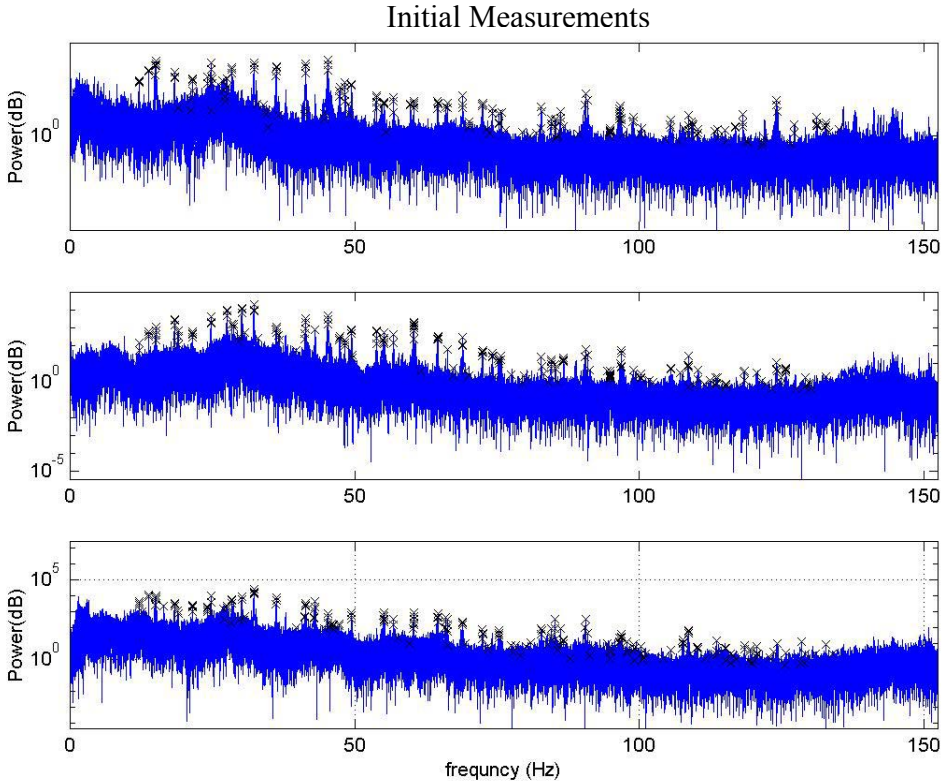
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

14-Lead CERPAK
(QC-14)



Appendix F: Initial Measurements

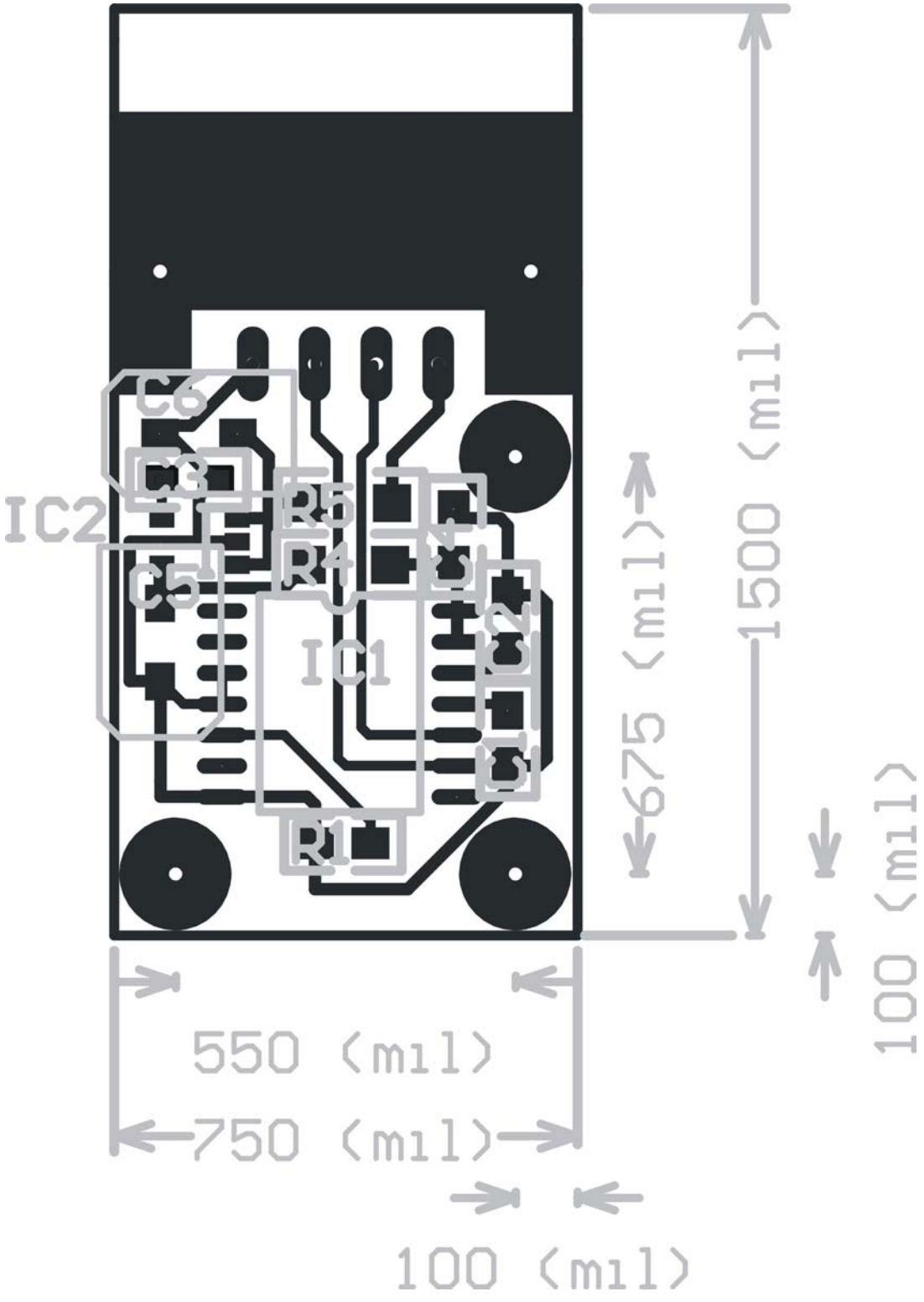


Lateral Direction

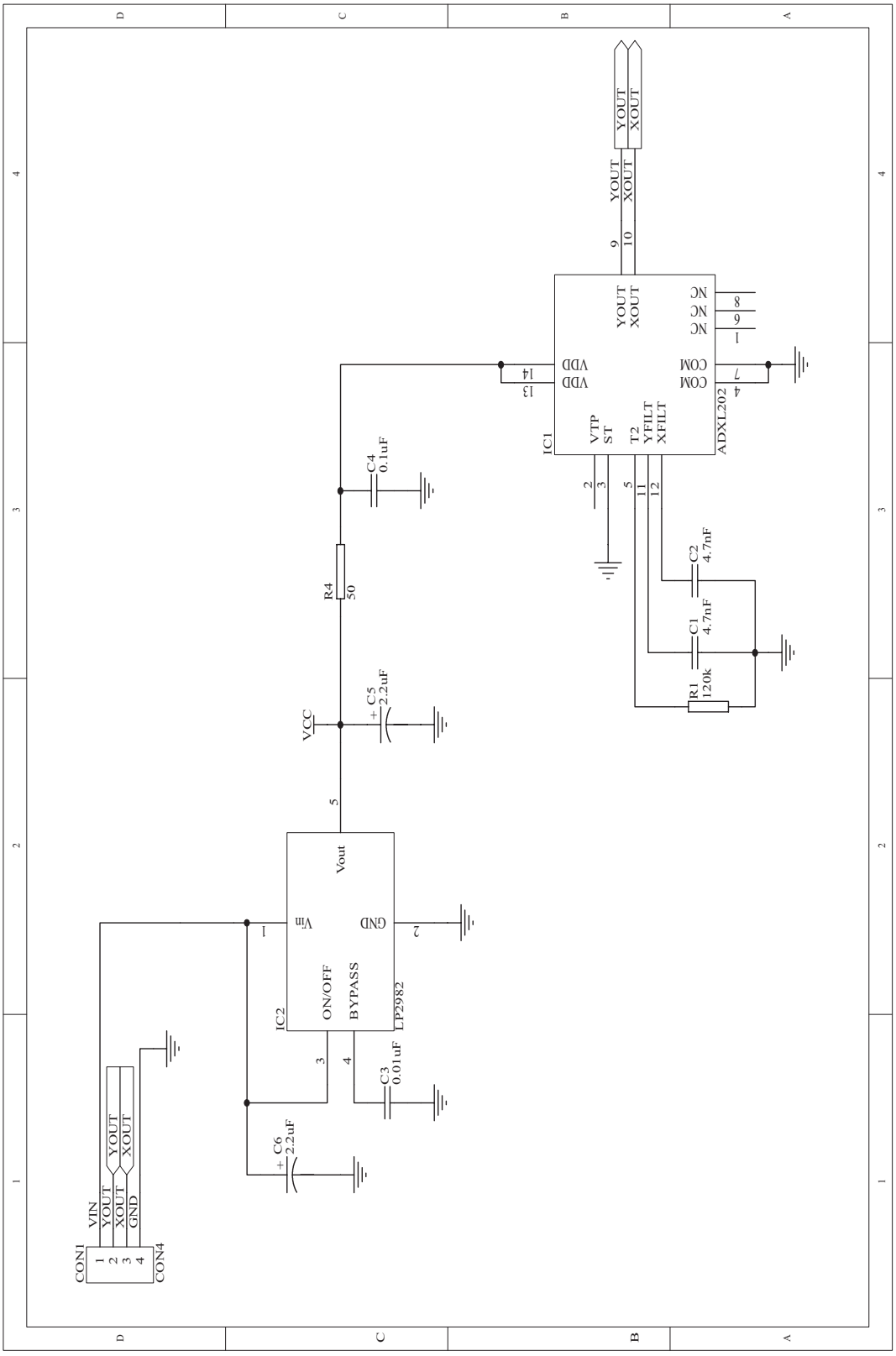
Longitudinal Direction

Vertical Direction

Appendix G: Accelerometers Printed Circuit-board



Appendix H: Schematic of Accelerometers.



Appendix I: Specification of the FPGA.

XILINX®

Spartan and Spartan-XL Families Field Programmable Gate Arrays

DS060 (v1.5) March 2, 2000 — Product Specification

Introduction

The Spartan™ series is the first high-volume production FPGA solution to deliver all the key requirements for ASIC replacement up to 40,000 gates. These requirements include high performance, on-chip RAM, core solutions and prices that, in high volume, approach and in many cases are equivalent to mask programmed ASIC devices.

The Spartan series is the result of more than 14 years of FPGA design experience and feedback from thousands of customers. By streamlining the Spartan series feature set, leveraging advanced hybrid process technologies and focusing on total cost management, the Spartan series delivers the key features required by ASIC and other high-volume logic users while avoiding the initial cost, long development cycles and inherent risk of conventional ASICs. The Spartan and Spartan-XL families in the Spartan series have ten members, as shown in Table 1.

Spartan and Spartan-XL Features

Note: The Spartan series devices described in this data sheet include the 5V Spartan family and the 3.3V Spartan-XL family. See the separate data sheet for the 2.5V Spartan-II family.

- First ASIC replacement FPGA for high-volume production with on-chip RAM
- Advanced process technology
- Density up to 1862 logic cells or 40,000 system gates
- Streamlined feature set based on XC4000 architecture
- System performance beyond 80 MHz
- Broad set of AllianceCORE™ and LogiCORE™ predefined solutions available
- Unlimited reprogrammability
- Low cost
- System level features
  - Available in both 5V and 3.3V versions
  - On-chip SelectRAM™ memory
  - Fully PCI compliant
  - Low power segmented routing architecture
  - Full readback capability for program verification and internal node observability
  - Dedicated high-speed carry logic
  - Internal 3-state bus capability
  - Eight global low-skew clock or signal networks
  - IEEE 1149.1-compatible Boundary Scan logic
- Versatile I/O and packaging
  - Low cost plastic packages available in all densities
  - Footprint compatibility in common packages
  - Individually programmable output slew-rate control maximizes performance and reduces noise
  - Zero input register hold time simplifies system timing
- Fully supported by powerful Xilinx development system
  - Foundation Series: Integrated, shrink-wrap software
  - Alliance Series: Dozens of PC and workstation third party development systems supported
  - Fully automatic mapping, placement and routing

Additional Spartan-XL Features

- 3.3V supply for low power with 5V tolerant I/Os
- Power down input
- Higher performance
- Faster carry logic
- More flexible high-speed clock network
- Latch capability in Configurable Logic Blocks
- Input fast capture latch
- Optional mux or 2-input function generator on outputs
- 12 mA or 24 mA output drive
- 5V and 3.3V PCI compatible
- Enhanced Boundary Scan
- Express Mode configuration
- Chip scale packaging

Table 1: Spartan and Spartan-XL Field Programmable Gate Arrays

Device	Logic Cells	Max System Gates	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total CLBs	Number of Flip-flops	Max. Available User I/O
XCS05 & XCS05XL	238	5,000	2,000 - 5,000	10 x 10	100	360	77
XCS10 & XCS10XL	466	10,000	3,000 - 10,000	14 x 14	196	616	112
XCS20 & XCS20XL	950	20,000	7,000 - 20,000	20 x 20	400	1,120	160
XCS30 & XCS30XL	1368	30,000	10,000 - 30,000	24 x 24	576	1,536	192
XCS40 & XCS40XL	1862	40,000	13,000 - 40,000	28 x 28	784	2,016	224

* Max values of Typical Gate Range include 20-30% of CLBs used as RAM.

General Overview

Spartan series FPGAs are implemented with a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), interconnected by a powerful hierarchy of versatile routing resources (routing channels), and surrounded by a perimeter of programmable Input/Output Blocks (IOBs), as seen in Figure 1. They have generous routing resources to accommodate the most complex interconnect patterns.

The devices are customized by loading configuration data into internal static memory cells. Re-programming is possible an unlimited number of times. The values stored in these memory cells determine the logic functions and interconnections implemented in the FPGA. The FPGA can either actively read its configuration data from an external serial PROM (Master Serial mode), or the configuration data can be written into the FPGA from an external device (Slave Serial mode).

Spartan series FPGAs can be used where hardware must be adapted to different user applications. FPGAs are ideal for shortening design and development cycles, and also offer a cost-effective solution for production rates well beyond 50,000 systems per month.

Spartan series devices achieve high-performance, low-cost operation through the use of an advanced architecture and semiconductor technology. Spartan and Spartan-XL devices provide system clock rates exceeding 80 MHz and internal performance in excess of 150 MHz. In contrast to other FPGA devices, the Spartan series offers the most cost-effective solution while maintaining leading-edge performance. In addition to the conventional benefit of high volume programmable logic solutions, Spartan series FPGAs also offer on-chip edge-triggered single-port and dual-port RAM, clock enables on all flip-flops, fast carry logic, and many other features.

The Spartan/XL families leverage the highly successful XC4000 architecture with many of that family's features and benefits. Technology advancements have been derived from the XC4000XLA process developments.

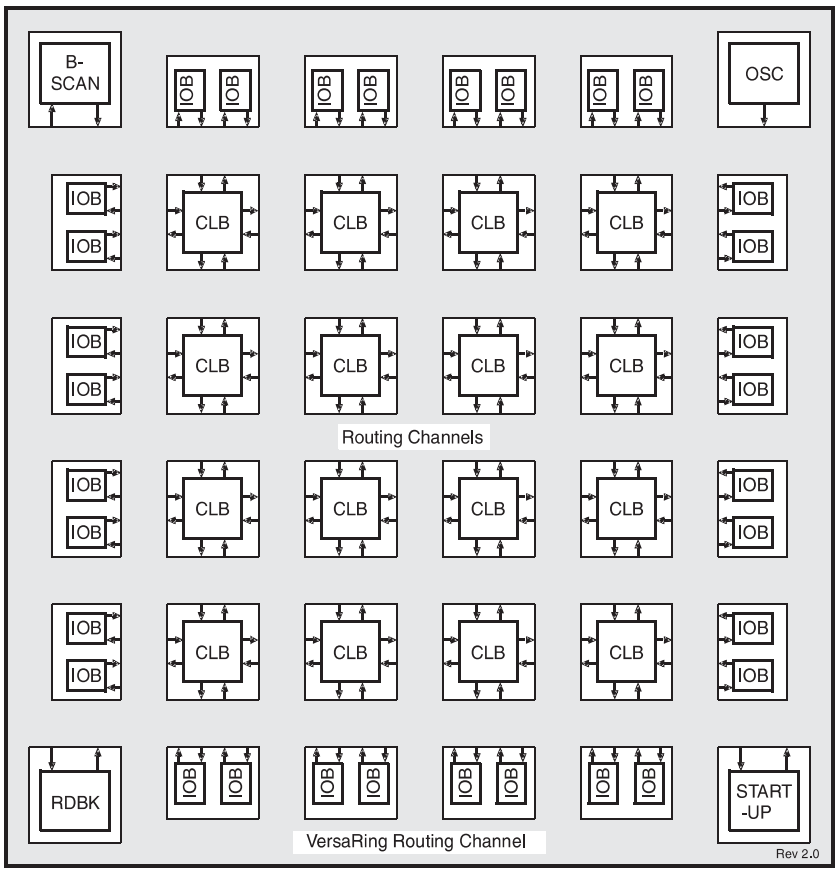


Figure 1: Basic FPGA Block Diagram

Logic Functional Description

The Spartan series uses a standard FPGA structure as shown in Figure 1 on page 2. The FPGA consists of an array of configurable logic blocks (CLBs) placed in a matrix of routing channels. The input and output of signals is achieved through a set of input/output blocks (IOBs) forming a ring around the CLBs and routing channels.

- CLBs provide the functional elements for implementing the user's logic.
- IOBs provide the interface between the package pins and internal signal lines.
- Routing channels provide paths to interconnect the inputs and outputs of the CLBs and IOBs.

The functionality of each circuit block is customized during configuration by programming internal static memory cells. The values stored in these memory cells determine the logic functions and interconnections implemented in the FPGA.

Configurable Logic Blocks (CLBs)

The CLBs are used to implement most of the logic in an FPGA. The principal CLB elements are shown in the simplified block diagram in Figure 2. There are three look-up tables (LUT) which are used as logic function generators, two flip-flops and two groups of signal steering multiplexers. There are also some more advanced features provided by the CLB which will be covered in the "Advanced Features Description" on page 12.

Function Generators

Two 16 x 1 memory look-up tables (F-LUT and G-LUT) are used to implement 4-input function generators, each offering unrestricted logic implementation of any Boolean function of up to four independent input signals (F1 to F4 or G1 to G4). Using memory look-up tables the propagation delay is independent of the function implemented.

A third 3-input function generator (H-LUT) can implement any Boolean function of its three inputs. Two of these inputs are controlled by programmable multiplexers (see box "A" of Figure 2). These inputs can come from the F-LUT or G-LUT outputs or from CLB inputs. The third input always comes from a CLB input. The CLB can, therefore, implement certain functions of up to nine inputs, like parity checking. The three LUTs in the CLB can also be combined to do any arbitrarily defined Boolean function of five inputs.

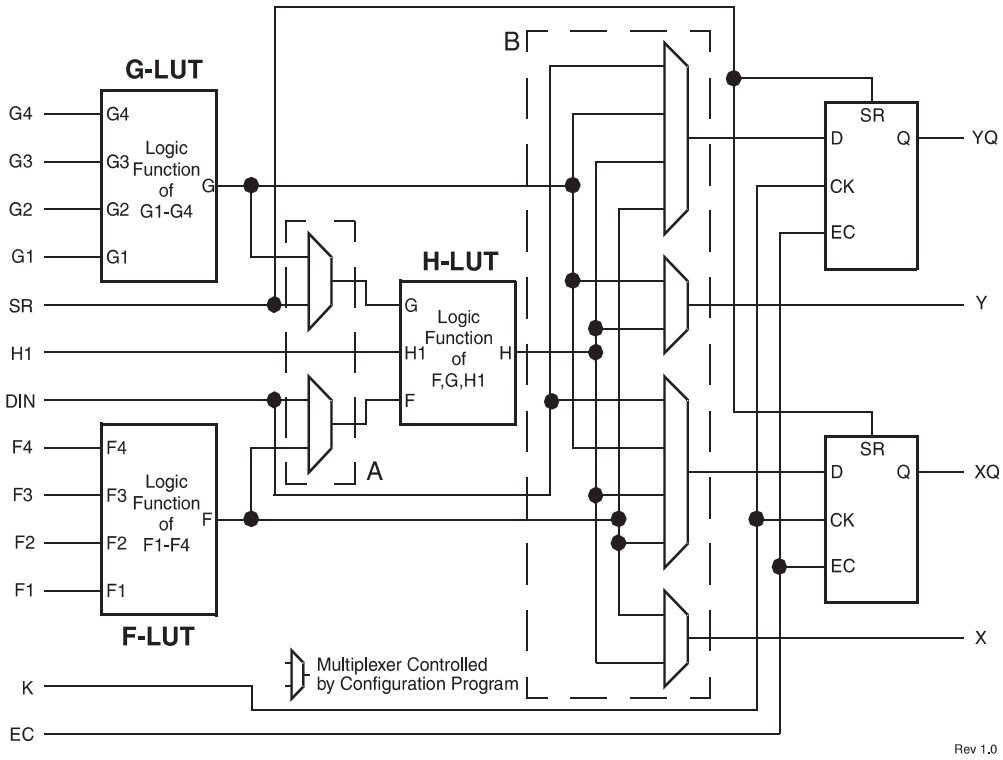


Figure 2: Spartan/XL Simplified CLB Logic Diagram (some features not shown)

A CLB can implement any of the following functions:

- Any function of up to four variables, plus any second function of up to four unrelated variables, plus any third function of up to three unrelated variables[1]
- Any single function of five variables
- Any function of four variables together with some functions of six variables
- Some functions of up to nine variables.

Implementing wide functions in a single block reduces both the number of blocks required and the delay in the signal path, achieving both increased capacity and speed.

The versatility of the CLB function generators significantly improves system speed. In addition, the design-software tools can deal with each function generator independently. This flexibility improves cell usage.

Flip-Flops

Each CLB contains two flip-flops that can be used to register (store) the function generator outputs. The flip-flops and function generators can also be used independently (see Figure 2 on page 3). The CLB input DIN can be used as a direct input to either of the two flip-flops. H1 can also drive either flip-flop via the H-LUT with a slight additional delay.

The two flip-flops have common clock (CK), clock enable (EC) and set/reset (SR) inputs. Internally both flip-flops are also controlled by a global initialization signal (GSR) which is described in detail in "Global Signals: GSR and GTS" on page 18.

Latches (Spartan-XL only)

The Spartan-XL CLB storage elements can also be configured as latches. The two latches have common clock (K) and clock enable (EC) inputs. Functionality of the storage element is described in Table 2.

Table 2: CLB Storage Element Functionality

Mode	CK	EC	SR	D	Q
Power-Up or GSR	X	X	X	X	SR
Flip-Flop Operation	X	X	1	X	SR
	⎍/‾	1*	0*	D	D
	0	X	0*	X	Q
Latch Operation (Spartan-XL)	1	1*	0*	X	Q
	0	1*	0*	D	D
Both	X	0	0*	X	Q

Legend:

X	Don't care
⎍/	Rising edge (clock not inverted)
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)

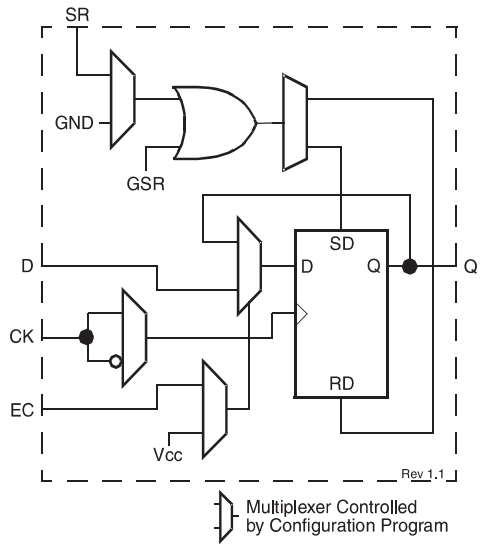


Figure 3: CLB Flip-Flop Functional Block Diagram

Clock Input

Each flip-flop can be triggered on either the rising or falling clock edge. The CLB clock line is shared by both flip-flops. However, the clock is individually invertible for each flip-flop (see CK path in Figure 3). Any inverter placed on the clock line in the design is automatically absorbed into the CLB.

Clock Enable

The clock enable line (EC) is active High. The EC line is shared by both flip-flops in a CLB. If either one is left disconnected, the clock enable for that flip-flop defaults to the active state. EC is not invertible within the CLB. The clock enable is synchronous to the clock and must satisfy the setup and hold timing specified for the device.

Set/Reset

The set/reset line (SR) is an asynchronous active High control of the flip-flop. SR can be configured as either set or reset at each flip-flop. This configuration option determines the state in which each flip-flop becomes operational after configuration. It also determines the effect of a GSR pulse during normal operation, and the effect of a pulse on the SR line of the CLB. The SR line is shared by both flip-flops. If SR is not specified for a flip-flop the set/reset for that flip-flop defaults to the inactive state. SR is not invertible within the CLB.

1. When three separate functions are generated, one of the function outputs must be captured in a flip-flop internal to the CLB. Only two unregistered function generator outputs are available from the CLB.

CLB Signal Flow Control

In addition to the H-LUT input control multiplexers (shown in box "A" of Figure 2 on page 3) there are signal flow control multiplexers (shown in box "B" of Figure 2) which select the signals which drive the flip-flop inputs and the combinatorial CLB outputs (X and Y).

Each flip-flop input is driven from a 4:1 multiplexer which selects among the three LUT outputs and DIN as the data source.

Each combinatorial output is driven from a 2:1 multiplexer which selects between two of the LUT outputs. The X output can be driven from the F-LUT or H-LUT, the Y output from G-LUT or H-LUT.

Control Signals

There are four signal control multiplexers on the input of the CLB. These multiplexers allow the internal CLB control signals (H1, DIN, SR, and EC in Figure 2 and Figure 4) to be driven from any of the four general control inputs (C1 - C4 in Figure 4) into the CLB. Any of these inputs can drive any of the four internal control signals.

The four internal control signals are:

- EC - Enable Clock
- SR - Asynchronous Set/Reset or H function generator Input 0
- DIN - Direct In or H function generator Input 2
- H1 - H function generator Input 1.

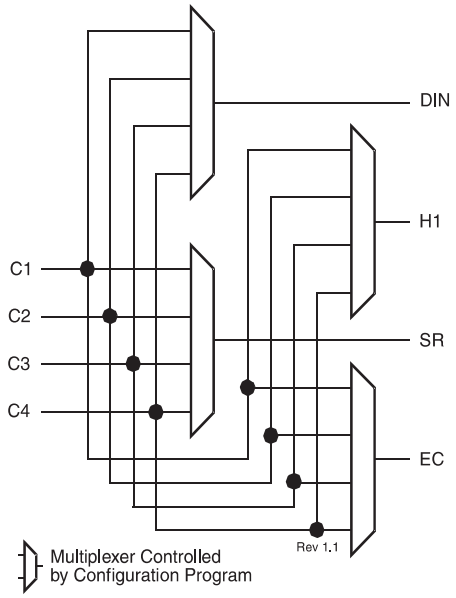


Figure 4: CLB Control Signal Interface

Input/Output Blocks (IOBs)

User-configurable input/output blocks (IOBs) provide the interface between external package pins and the internal logic. Each IOB controls one package pin and can be configured for input, output, or bidirectional signals. Figure 5 on page 6 shows a simplified functional block diagram of the Spartan/XL IOB.

IOB Input Signal Path

The input signal to the IOB can be configured to either go directly to the routing channels (via I1 and I2 in Figure 5) or to the input register. The input register can be programmed as either an edge-triggered flip-flop or a level-sensitive latch. The functionality of this register is shown in Table 3, and a simplified block diagram of the register can be seen in Figure 6.

Table 3: Input Register Functionality

Mode	CK	EC	D	Q
Power-Up or GSR	X	X	X	SR
Flip-Flop	_/‾	1*	D	D
	0	X	X	Q
Latch	1	1*	X	Q
	0	1*	D	D
Both	X	0	X	Q

Legend:

X	Don't care
_/‾	Rising edge (clock not inverted)
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)

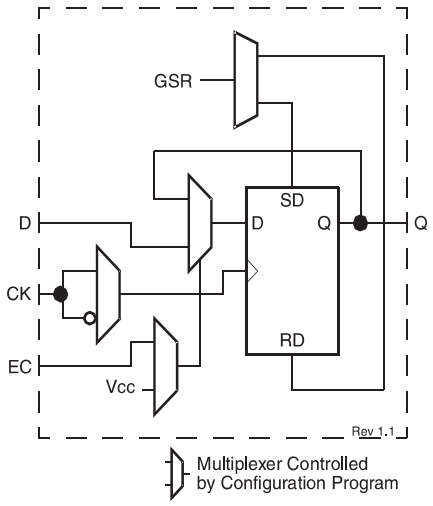


Figure 6: IOB Flip-Flop/Latch Functional Block Diagram

The register choice is made by placing the appropriate library symbol. For example, IFD is the basic input flip-flop (rising edge triggered), and ILD is the basic input latch (transparent-High). Variations with inverted clocks are also available. The clock signal inverter is also shown in Figure 6 on the CK line.

The Spartan IOB data input path has a one-tap delay element: either the delay is inserted (default), or it is not. The Spartan-XL IOB data input path has a two-tap delay element, with choices of a full delay, a partial delay, or no delay. The added delay guarantees a zero hold time with respect to clocks routed through the global clock buffers. (See "Global Nets and Buffers" on page 11 for a description of the global clock buffers in the Spartan/XL families.) For a shorter input register setup time, with positive hold-time, attach a NODELAY attribute or property to the flip-flop.

The output of the input register goes to the routing channels (via I1 and I2 in Figure 5). The I1 and I2 signals that exit the IOB can each carry either the direct or registered input signal.

The 5V Spartan input buffers can be globally configured for either TTL (1.2V) or CMOS (VCC/2) thresholds, using an option in the bitstream generation software. The Spartan output levels are also configurable; the two global adjustments of input threshold and output level are independent. The inputs of Spartan devices can be driven by the outputs of any 3.3V device, if the Spartan inputs are in TTL mode. Spartan-XL inputs are TTL compatible and 3.3V CMOS compatible.

Supported sources for Spartan/XL device inputs are shown in Table 4.

Spartan-XL I/Os are fully 5V tolerant even though the V_{CC} is 3.3V. This allows 5V signals to directly connect to the Spartan-XL inputs without damage, as shown in Table 4. In addition, the 3.3V V_{CC} can be applied before or after 5V signals are applied to the I/Os. This makes the Spartan-XL devices immune to power supply sequencing problems.

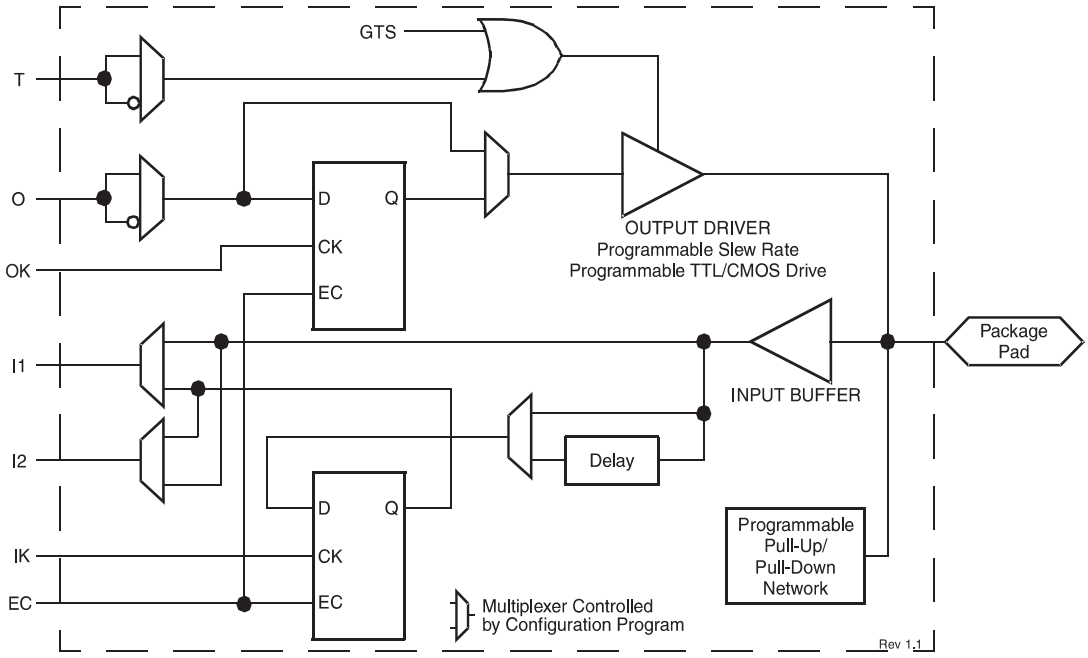


Figure 5: Simplified Spartan/XL IOB Block Diagram

Table 4: Supported Sources for Spartan/XL Inputs

Source	Spartan Inputs		Spartan-XL Inputs
	5V, TTL	5V, CMOS	3.3V CMOS
Any device, V_{CC} = 3.3V, CMOS outputs	√	Unreliable Data	√
Spartan family, V_{CC} = 5V, TTL outputs	√		√
Any device, V_{CC} = 5V, TTL outputs ($V_{OH} \leq 3.7V$)	√		√
Any device, V_{CC} = 5V, CMOS outputs	√	√	√ (default mode)

Spartan-XL V_{CC} Clamping

Spartan-XL FPGAs have an optional clamping diode connected from each I/O to V_{CC}. When enabled they clamp ringing transients back to the 3.3V supply rail. This clamping action is required in 3.3V PCI applications. V_{CC} clamping is a global option affecting all I/O pins.

Spartan-XL devices are fully 5V TTL I/O compatible if V_{CC} clamping is not enabled. With V_{CC} clamping enabled, the Spartan-XL devices will begin to clamp input voltages to one diode voltage drop above V_{CC}. If enabled, TTL I/O compatibility is maintained but full 5V I/O tolerance is sacrificed. The user may select either 5V tolerance (default) or 3.3V PCI compatibility. In both cases negative voltage is clamped to one diode voltage drop below ground.

Spartan-XL devices are compatible with TTL, LVTTL, PCI 3V, PCI 5V and LVCMOS signalling. The various standards are illustrated in Table 5.

Table 5: I/O Standards Supported by Spartan-XL FPGAs

Signaling Standard	VCC Clamping	Output Drive	$V_{IH\ MAX}$	$V_{IH\ MIN}$	$V_{IL\ MAX}$	$V_{OH\ MIN}$	$V_{OL\ MAX}$
TTL	Not allowed	12/24 mA	5.5	2.0	0.8	2.4	0.4
LVTTL	OK	12/24 mA	3.6	2.0	0.8	2.4	0.4
PCI5V	Not allowed	24 mA	5.5	2.0	0.8	2.4	0.4
PCI3V	Required	12 mA	3.6	50% of V_{CC}	30% of V_{CC}	90% of V_{CC}	10% of V_{CC}
LVCMOS 3V	OK	12/24 mA	3.6	50% of V_{CC}	30% of V_{CC}	90% of V_{CC}	10% of V_{CC}

Additional Fast Capture Input Latch (Spartan-XL only)

The Spartan-XL IOB has an additional optional latch on the input. This latch is clocked by the clock used for the output flip-flop rather than the input clock. Therefore, two different clocks can be used to clock the two input storage elements. This additional latch allows the fast capture of input data, which is then synchronized to the internal clock by the IOB flip-flop or latch.

To place the Fast Capture latch in a design, use one of the special library symbols, ILFFX or ILFLX. ILFFX is a transparent-Low Fast Capture latch followed by an active High input flip-flop. ILFLX is a transparent-Low Fast Capture latch followed by a transparent-High input latch. Any of the clock inputs can be inverted before driving the library element, and the inverter is absorbed into the IOB.

IOB Output Signal Path

Output signals can be optionally inverted within the IOB, and can pass directly to the output buffer or be stored in an edge-triggered flip-flop and then to the output buffer. The functionality of this flip-flop is shown in Table 6.

Table 6: Output Flip-Flop Functionality

Mode	Clock	Clock Enable	T	D	Q
Power-Up or GSR	X	X	0*	X	SR
Flip-Flop	X	0	0*	X	Q
	⎍/‾	1*	0*	D	D
	X	X	1	X	Z
	0	X	0*	X	Q

Legend:

X	Don't care
⎍/‾	Rising edge (clock not inverted)
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)
Z	3-state

Output Multiplexer/2-Input Function Generator (Spartan-XL only)

The output path in the Spartan-XL IOB contains an additional multiplexer not available in the Spartan IOB. The multiplexer can also be configured as a 2-input function generator, implementing a pass gate, AND gate, OR gate, or XOR gate, with 0, 1, or 2 inverted inputs.

When configured as a multiplexer, this feature allows two output signals to time-share the same output pad, effectively doubling the number of device outputs without requiring a larger, more expensive package. The select input is the pin used for the output flip-flop clock, OK.

When the multiplexer is configured as a 2-input function generator, logic can be implemented within the IOB itself. Combined with a Global buffer, this arrangement allows very high-speed gating of a single signal. For example, a wide decoder can be implemented in CLBs, and its output gated with a Read or Write Strobe driven by a global buffer.

The user can specify that the IOB function generator be used by placing special library symbols beginning with the letter "O." For example, a 2-input AND gate in the IOB function generator is called OAND2. Use the symbol input pin labeled "F" for the signal on the critical path. This signal is placed on the OK pin — the IOB input with the shortest delay to the function generator. Two examples are shown in Figure 7.

Figure 7: AND & MUX Symbols in Spartan-XL IOB

Output Buffer

An active High 3-state signal can be used to place the output buffer in a high-impedance state, implementing 3-state outputs or bidirectional I/O. Under configuration control, the output (O) and output 3-state (T) signals can be inverted. The polarity of these signals is independently configured for each IOB (see Figure 5 on page 6). An output can be configured as open-drain (open-collector) by tying the 3-state pin (T) to the output signal, and the input pin (I) to Ground.

By default, a 5V Spartan device output buffer pull-up structure is configured as a TTL-like totem-pole. The High driver is an n-channel pull-up transistor, pulling to a voltage one transistor threshold below V_{CC}. Alternatively, the outputs can be globally configured as CMOS drivers, with additional p-channel pull-up transistors pulling to V_{CC}. This option, applied using the bitstream generation software, applies to all outputs on the device. It is not individually programmable.

All Spartan-XL device outputs are configured as CMOS drivers, therefore driving rail-to-rail. The Spartan-XL outputs are individually programmable for 12 mA or 24 mA output drive.

Any 5V Spartan device with its outputs configured in TTL mode can drive the inputs of any typical 3.3V device. Supported destinations for Spartan/XL device outputs are shown in Table 7.

Three-State Register (Spartan-XL Only)

Spartan-XL devices incorporate an optional register controlling the three-state enable in the IOBs. The use of the three-state control register can significantly improve output enable and disable time.

Output Slew Rate

The slew rate of each output buffer is, by default, reduced, to minimize power bus transients when switching non-critical signals. For critical signals, attach a FAST attribute or property to the output buffer or flip-flop.

Spartan/XL devices have a feature called "Soft Start-up," designed to reduce ground bounce when all outputs are turned on simultaneously at the end of configuration. When the configuration process is finished and the device starts up, the first activation of the outputs is automatically slew-rate limited. Immediately following the initial activation of the I/O, the slew rate of the individual outputs is determined by the individual configuration option for each IOB.

Pull-up and Pull-down Network

Programmable pull-up and pull-down resistors are used for tying unused pins to V_{CC} or Ground to minimize power consumption and reduce noise sensitivity. The configurable pull-up resistor is a p-channel transistor that pulls to V_{CC}. The configurable pull-down resistor is an n-channel transistor that pulls to Ground. The value of these resistors is typically 20 kΩ – 100 kΩ (See "Spartan DC Characteristics Over Operating Conditions" on page 36.). This high value makes them unsuitable as wired-AND pull-up resistors.

Table 7: Supported Destinations for Spartan/XL Outputs

Destination	Spartan-XL Outputs	Spartan Outputs	
	3.3V, CMOS	5V, TTL	5V, CMOS
Any device, V_{CC} = 3.3V, CMOS-threshold inputs	√	√	Some[1]
Any device, V_{CC} = 5V, TTL-threshold inputs	√	√	√
Any device, V_{CC} = 5V, CMOS-threshold inputs	**Unreliable Data**		√

1. Only if destination device has 5V tolerant inputs

After configuration, voltage levels of unused pads, bonded or unbonded, must be valid logic levels, to reduce noise sensitivity and avoid excess current. Therefore, by default, unused pads are configured with the internal pull-up resistor active. Alternatively, they can be individually configured with the pull-down resistor, or as a driven output, or to be driven by an external source. To activate the internal pull-up, attach the PULLUP library component to the net attached to the pad. To activate the internal pull-down, attach the PULLDOWN library component to the net attached to the pad.

Set/Reset

As with the CLB registers, the GSR signal can be used to set or clear the input and output registers, depending on the value of the INIT attribute or property. The two flip-flops can be individually configured to set or clear on reset and after configuration. Other than the global GSR net, no user-controlled set/reset signal is available to the I/O flip-flops (Figure 6). The choice of set or reset applies to both the initial state of the flip-flop and the response to the GSR pulse.

Independent Clocks

Separate clock signals are provided for the input (IK) and output (OK) flip-flops. The clock can be independently inverted for each flip-flop within the IOB, generating either falling-edge or rising-edge triggered flip-flops. The clock inputs for each IOB are independent.

Common Clock Enables

The input and output flip-flops in each IOB have a common clock enable input (see EC signal in Figure 6), which through configuration, can be activated individually for the input or output flip-flop, or both. This clock enable operates exactly like the EC signal on the Spartan/XL CLB. It cannot be inverted within the IOB.

Routing Channel Description

All internal routing channels are composed of metal segments with programmable switching points and switching matrices to implement the desired routing. A structured, hierarchical matrix of routing channels is provided to achieve efficient automated routing.

This section describes the routing channels available in Spartan/XL devices. Figure 8 shows a general block diagram of the CLB routing channels. The implementation software automatically assigns the appropriate resources based on the density and timing requirements of the design. The following description of the routing channels is for information only and is simplified with some minor details omitted. For an exact interconnect description the designer should open a design in the FPGA Editor and review the actual connections in this tool.

The routing channels will be discussed as follows;

- CLB routing channels which run along each row and column of the CLB array.
- IOB routing channels which form a ring (called a VersaRing) around the outside of the CLB array. It connects the I/O with the CLB routing channels.
- Global routing consists of dedicated networks primarily designed to distribute clocks throughout the device with minimum delay and skew. Global routing can also be used for other high-fanout signals.

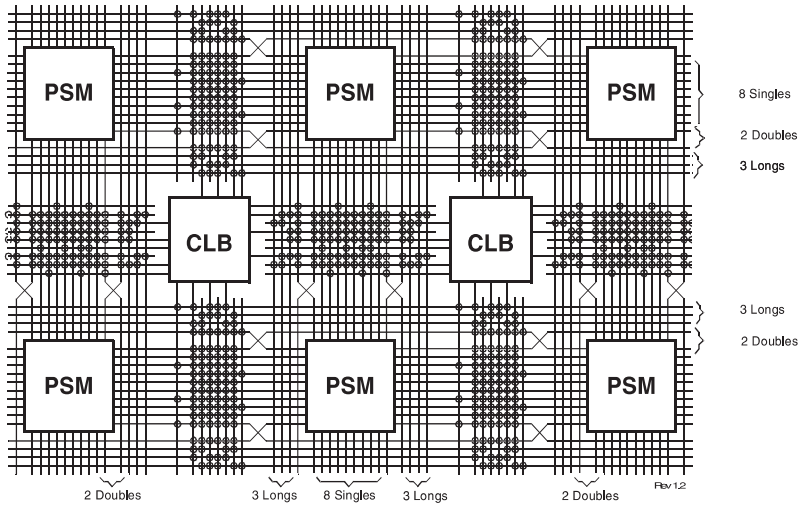


Figure 8: Spartan/XL CLB Routing Channels and Interface Block Diagram

CLB Routing Channels

The routing channels around the CLB are derived from three types of interconnects; single-length, double-length, and longlines. At the intersection of each vertical and horizontal routing channel is a signal steering matrix called a Programmable Switch Matrix (PSM). Figure 8 shows the basic routing channel configuration showing single-length lines, double-length lines and longlines as well as the CLBs and PSMs. The CLB to routing channel interface is shown as well as how the PSMs interface at the channel intersections.

CLB Interface

A block diagram of the CLB interface signals is shown in Figure 9. The input signals to the CLB are distributed evenly on all four sides providing maximum routing flexibility. In general, the entire architecture is symmetrical and regular. It is well suited to established placement and routing algorithms. Inputs, outputs, and function generators can freely swap positions within a CLB to avoid routing congestion during the placement and routing operation. The exceptions are the clock (K) input and CIN/COUT signals. The K input is routed to dedicated global vertical lines as well as four single-length lines and is on the left side of the CLB. The CIN/COUT signals are routed through dedicated interconnects which do not interfere with the general routing structure. The output signals from the CLB are available to drive both vertical and horizontal channels.

Programmable Switch Matrices

The horizontal and vertical single- and double-length lines intersect at a box called a programmable switch matrix (PSM). Each PSM consists of programmable pass transistors used to establish connections between the lines (see Figure 10).

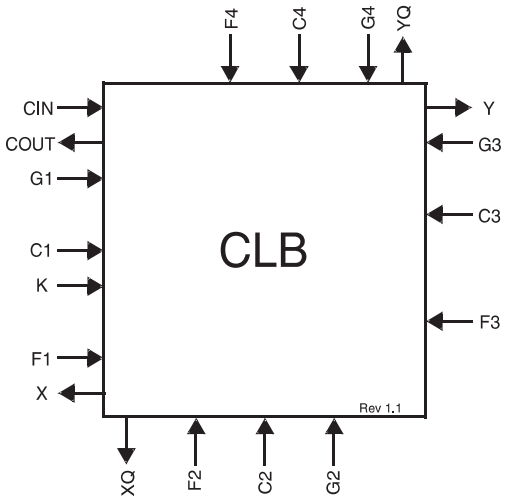


Figure 9: CLB Interconnect Signals

For example, a single-length signal entering on the right side of the switch matrix can be routed to a single-length line on the top, left, or bottom sides, or any combination thereof, if multiple branches are required. Similarly, a double-length signal can be routed to a double-length line on any or all of the other three edges of the programmable switch matrix.

Single-Length Lines

Single-length lines provide the greatest interconnect flexibility and offer fast routing between adjacent blocks. There are eight vertical and eight horizontal single-length lines associated with each CLB. These lines connect the switching matrices that are located in every row and column of CLBs.

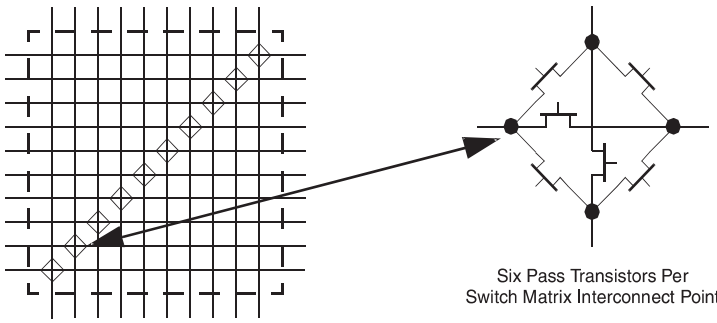


Figure 10: Programmable Switch Matrix

Appendix J: FPGA Printed Circuit-board Diagram.

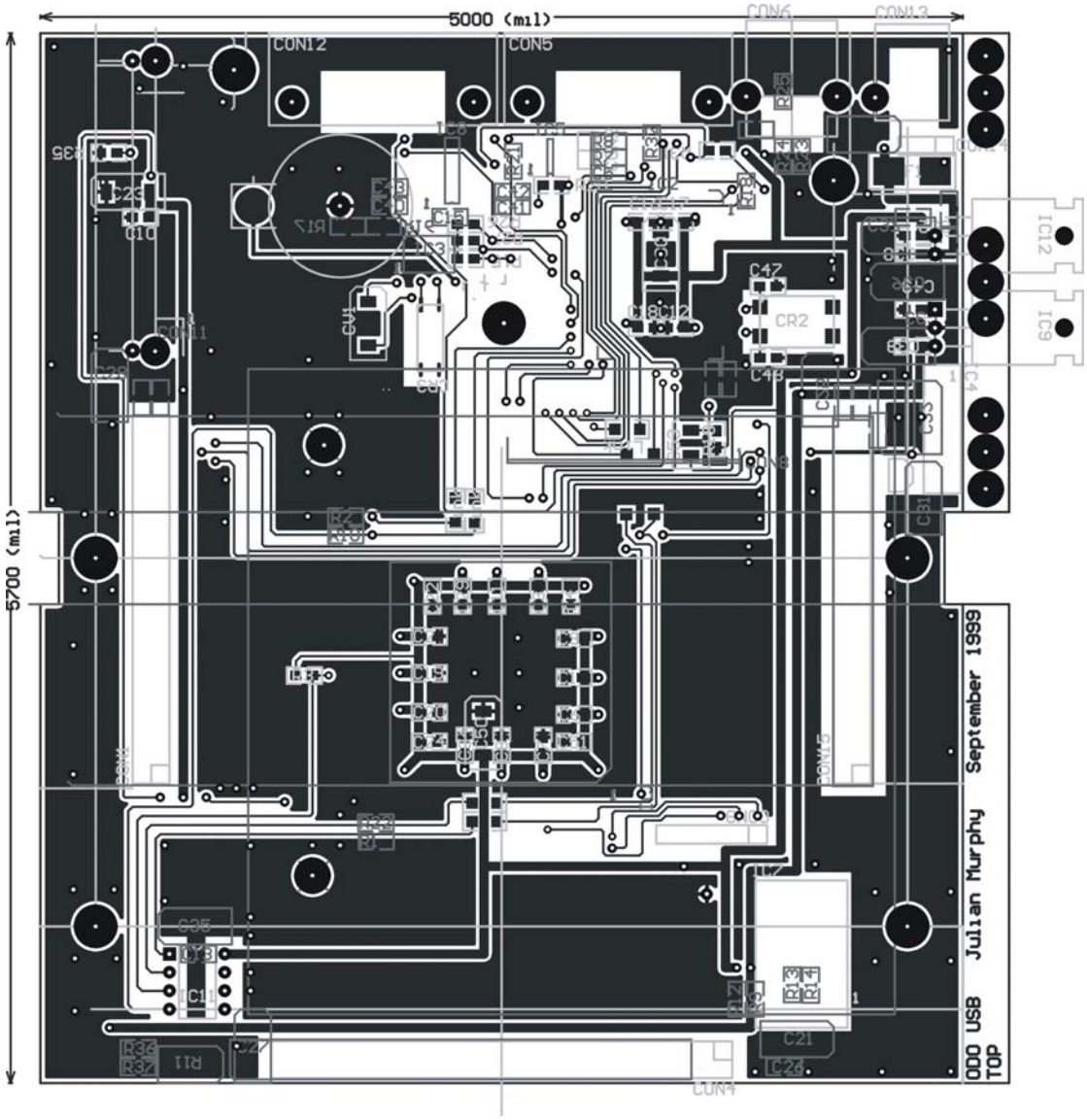


Figure J1, Top of the FPGA printed circuit-board.

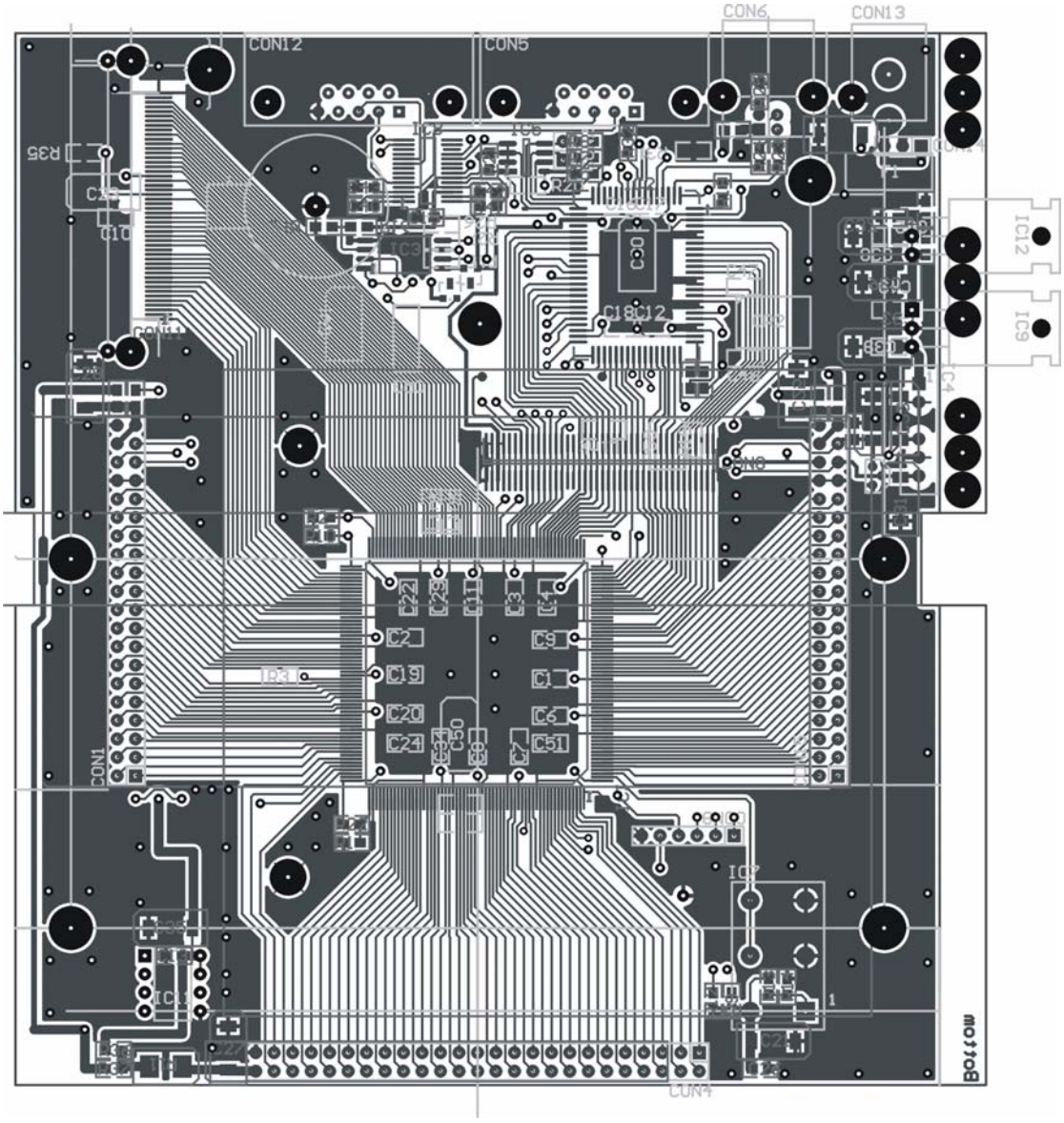


Figure J2, Bottom of the FPGA printed circuit-board.

Appendix K: Schematics for the FPGA Programmable Hardware

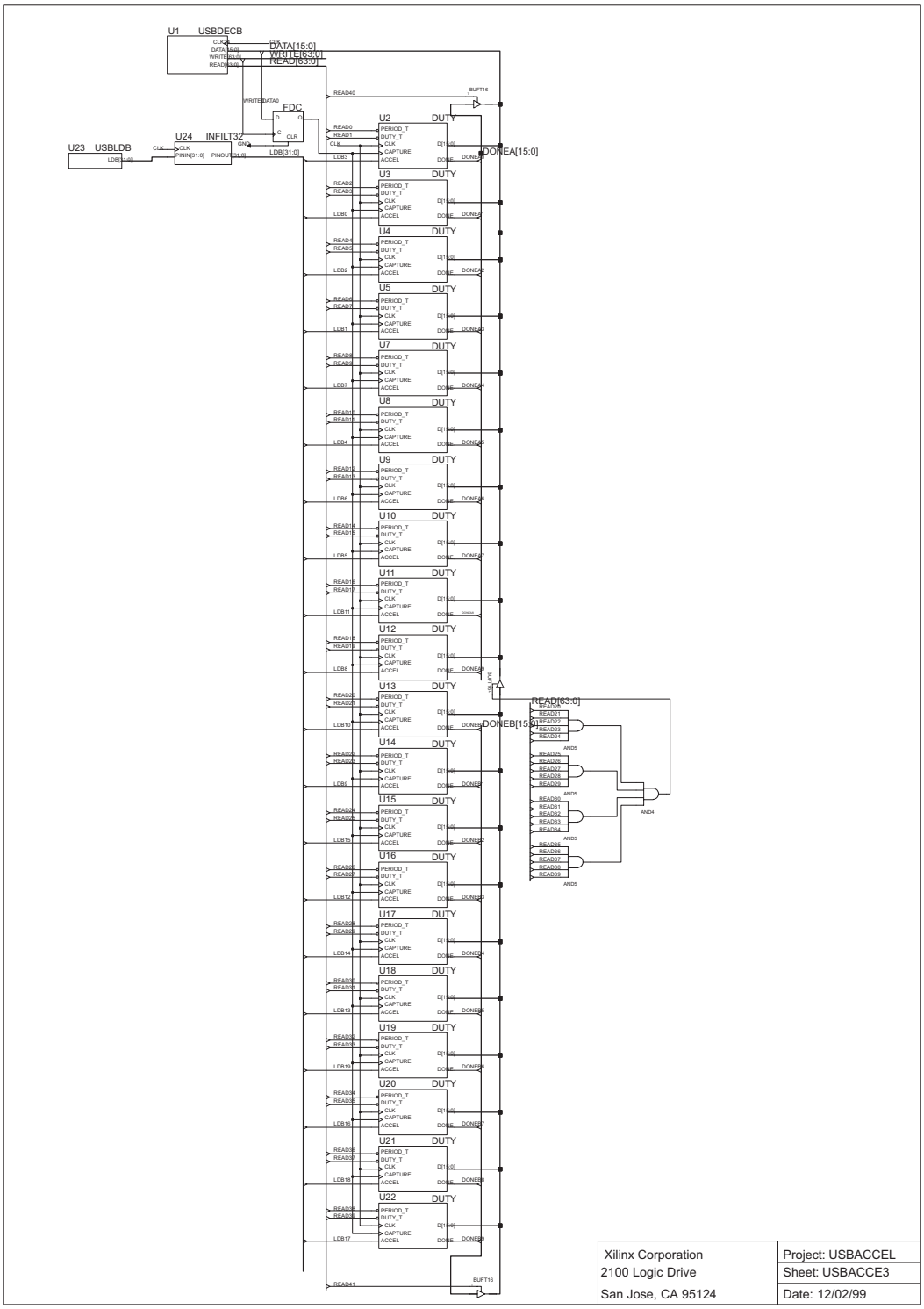


Figure K1, The overall schematic of the FPGA programmable hardware.

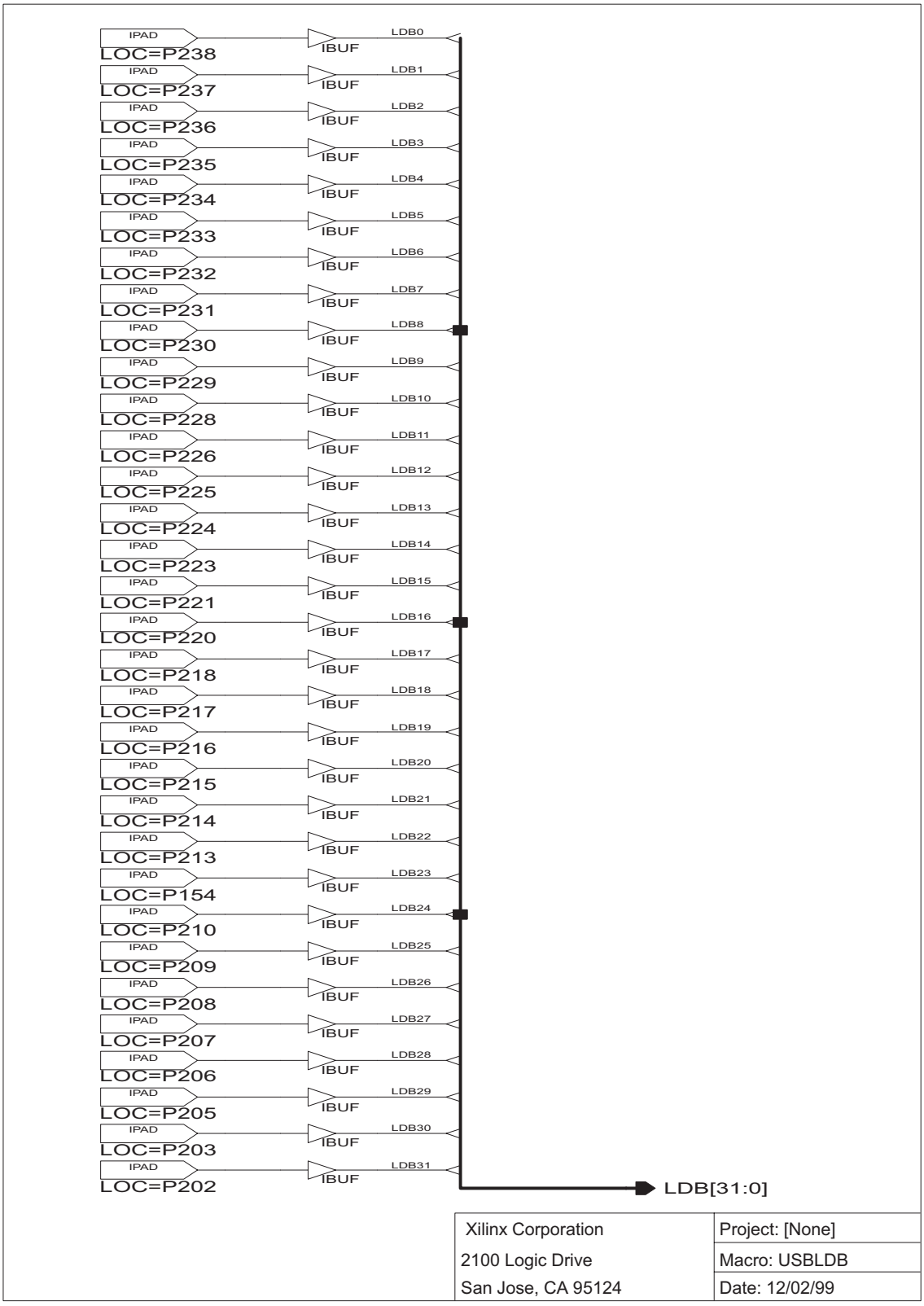


Figure K2, The detailed schematic of USBLDB.

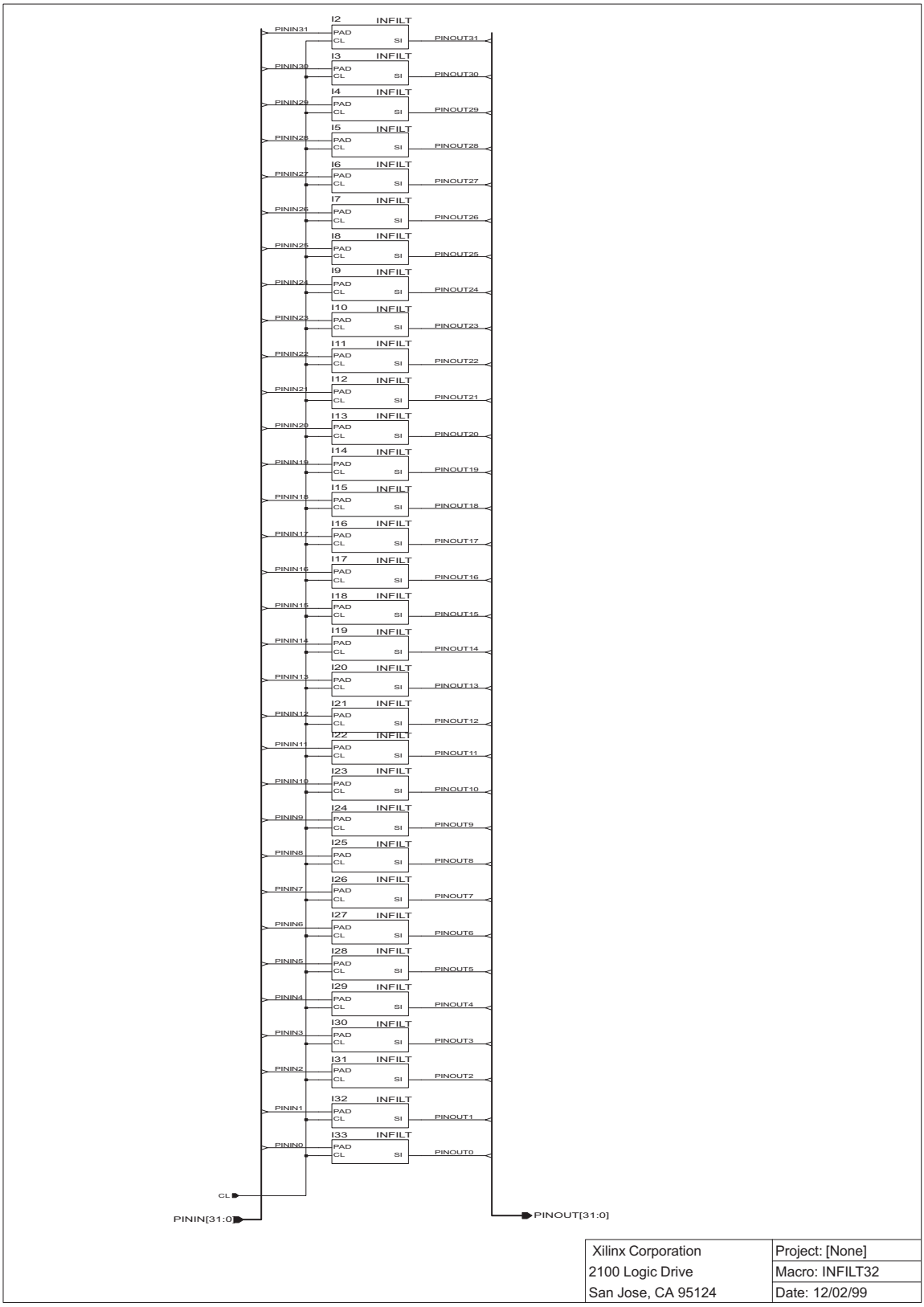


Figure K3, The detailed schematic INFILT32

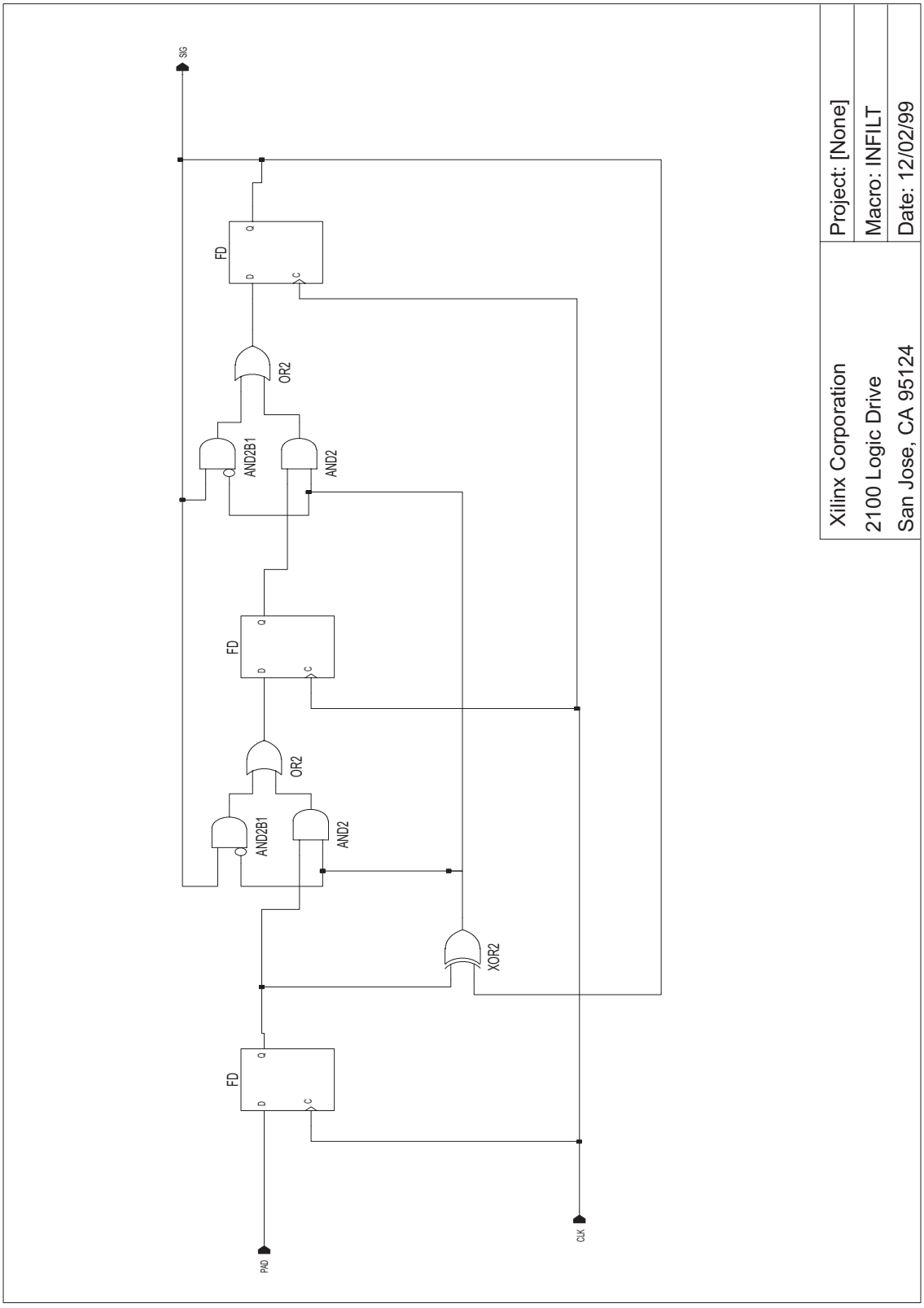


Figure K4, Detailed schematic of INFILT

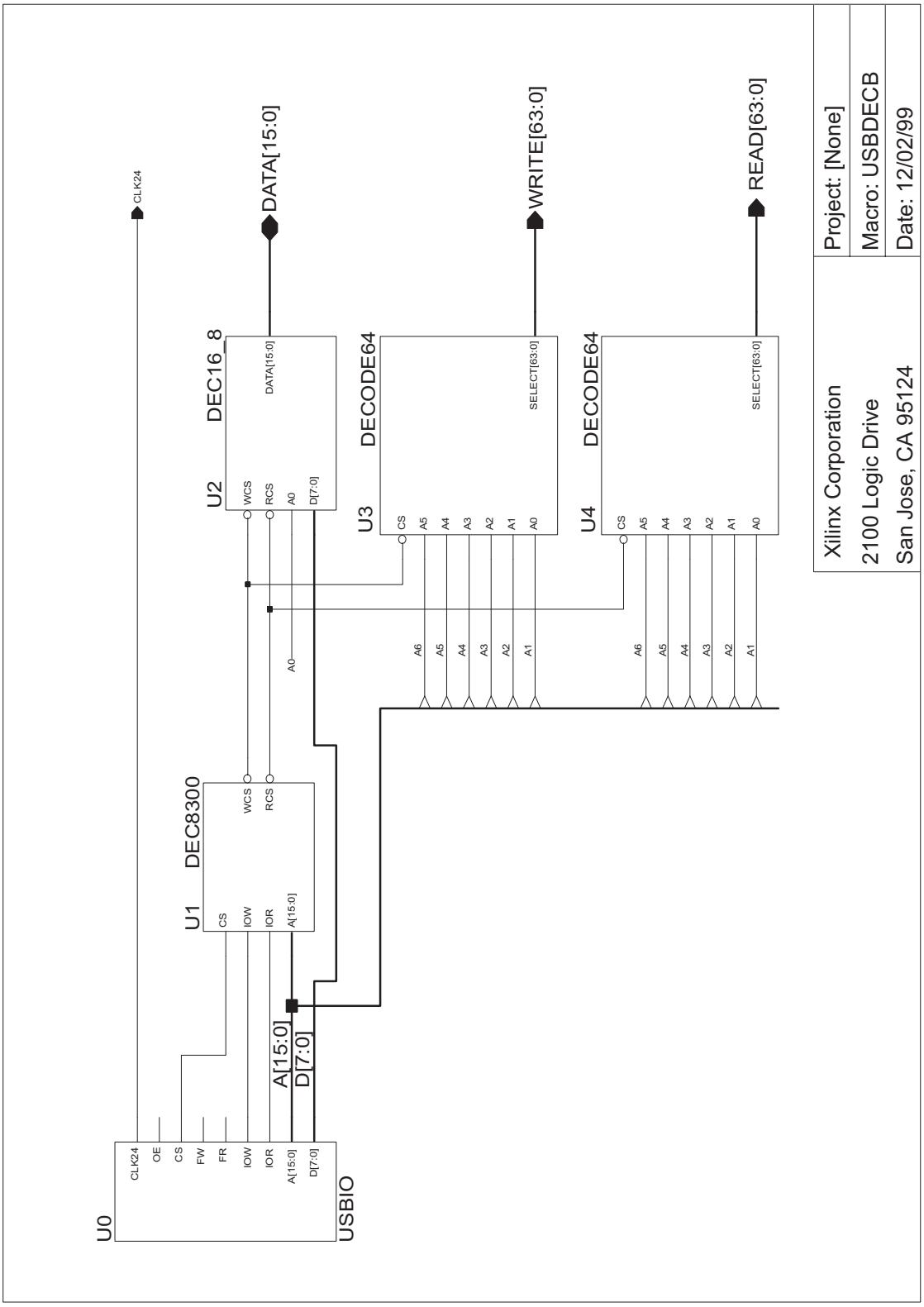


Figure K5, Detailed schematic of USBDECB

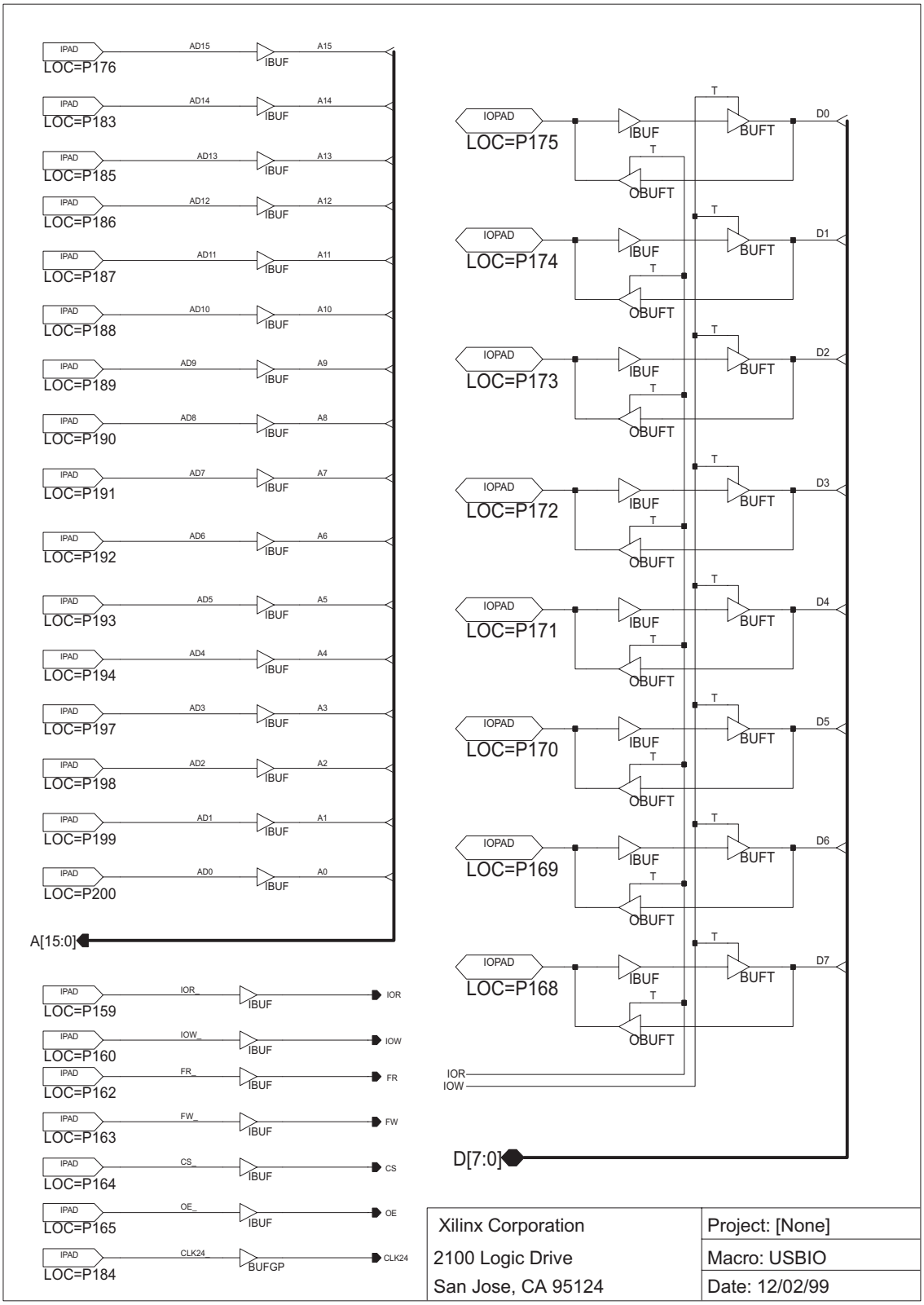


Figure K6, Detailed schematic of USBIO.

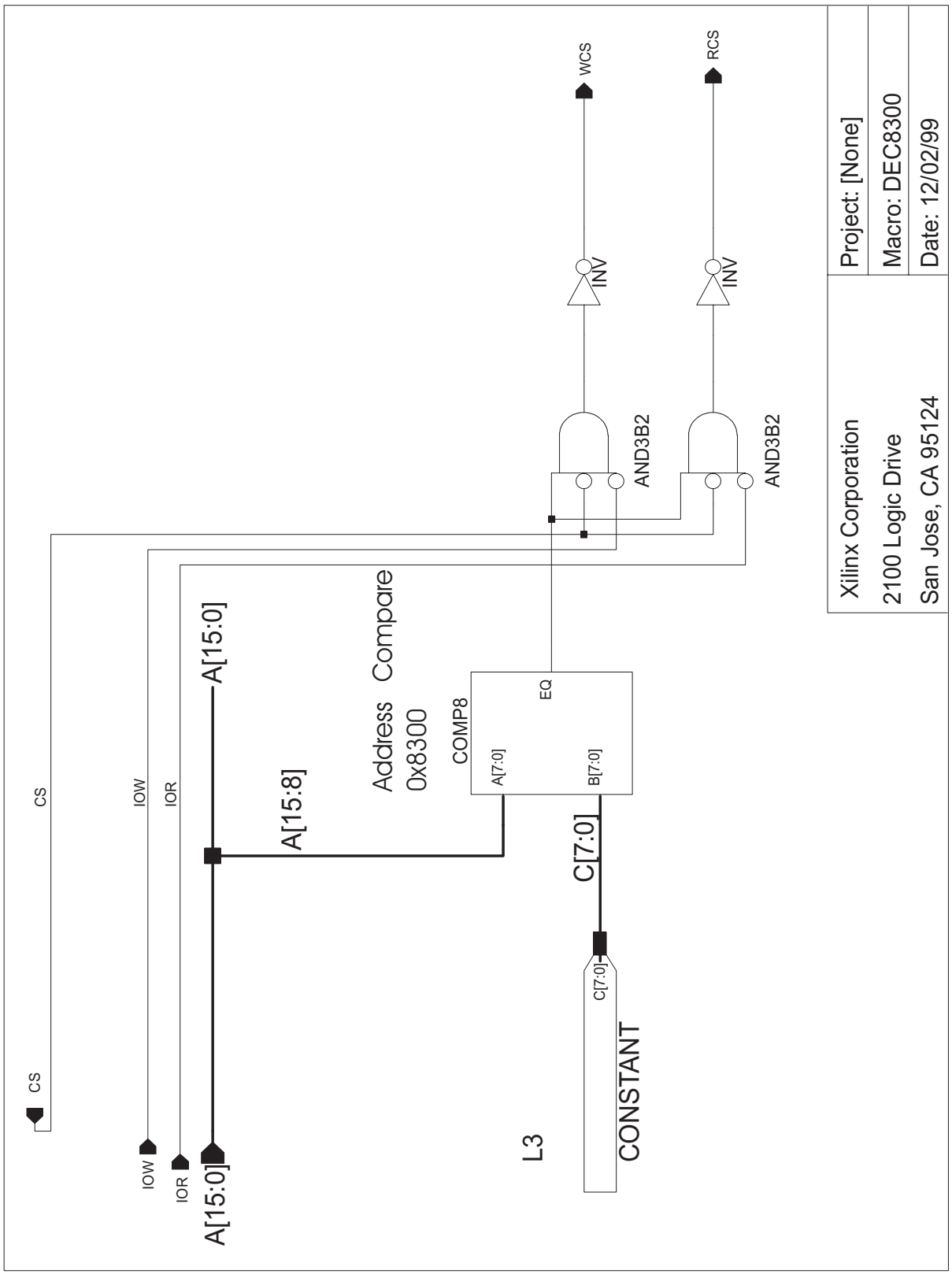


Figure K7, Detailed schematic DEC8300.

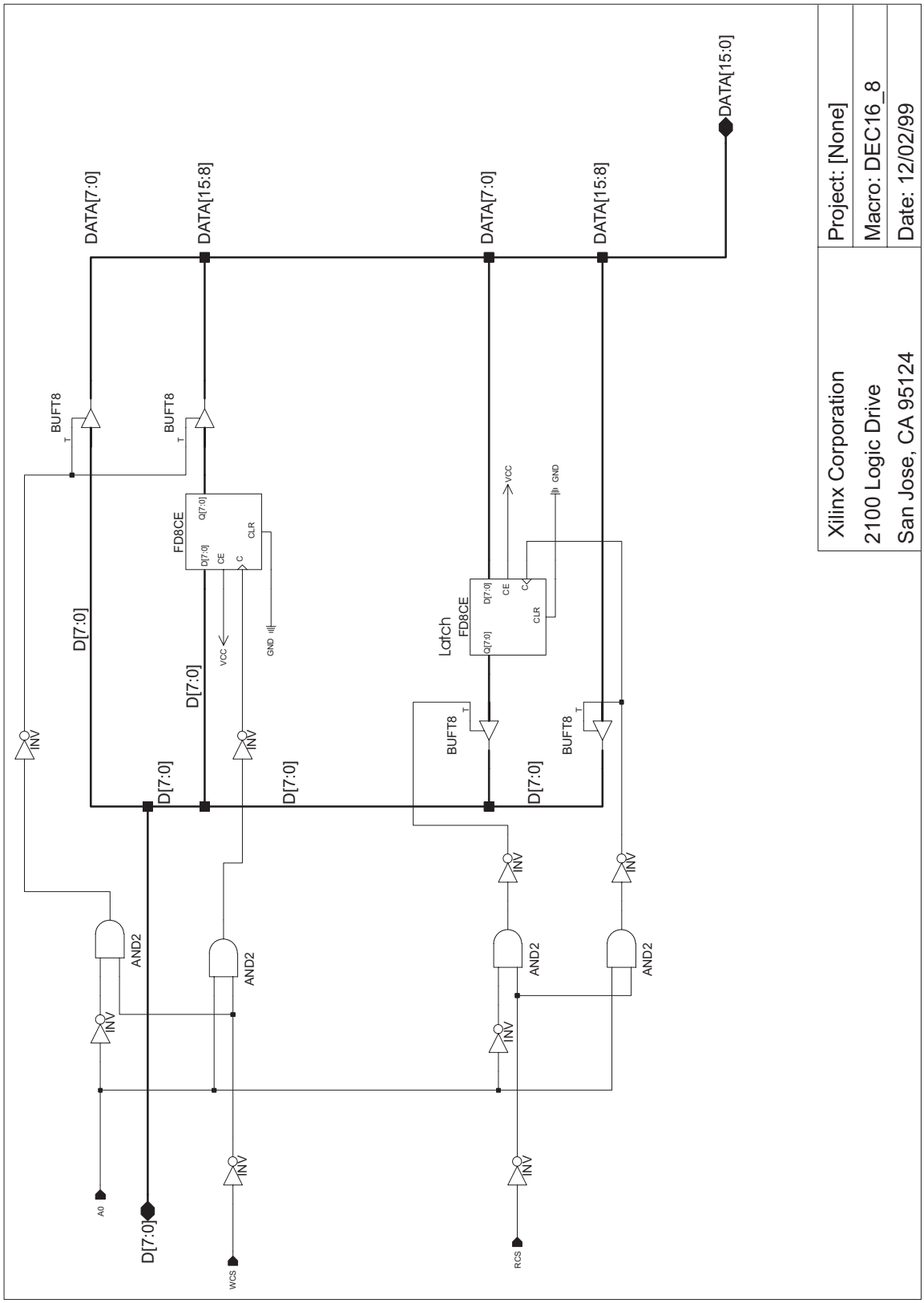


Figure K8, Detailed Schematic of DEC16_8.

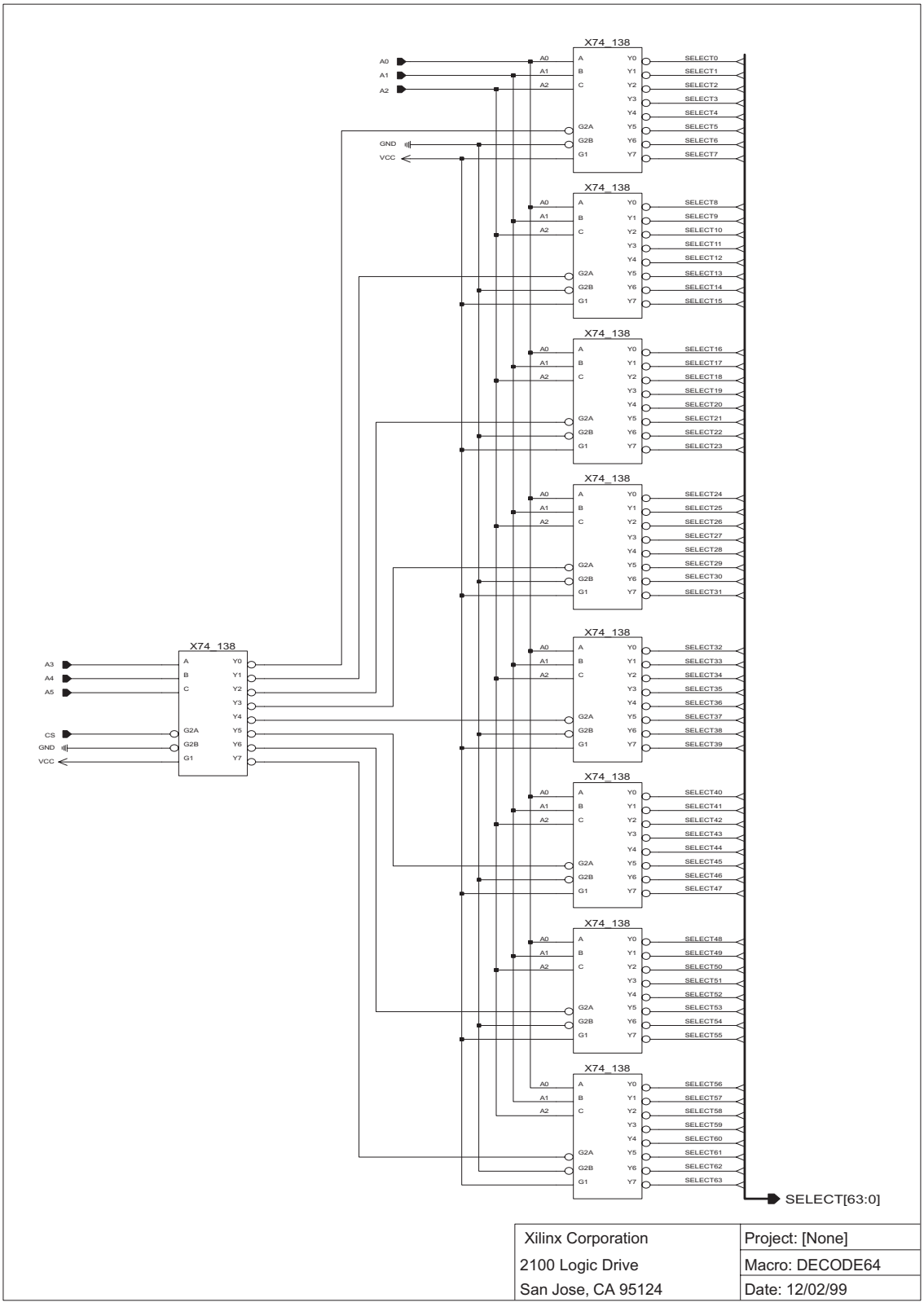


Figure K9, Detailed schematic of DECODE64.

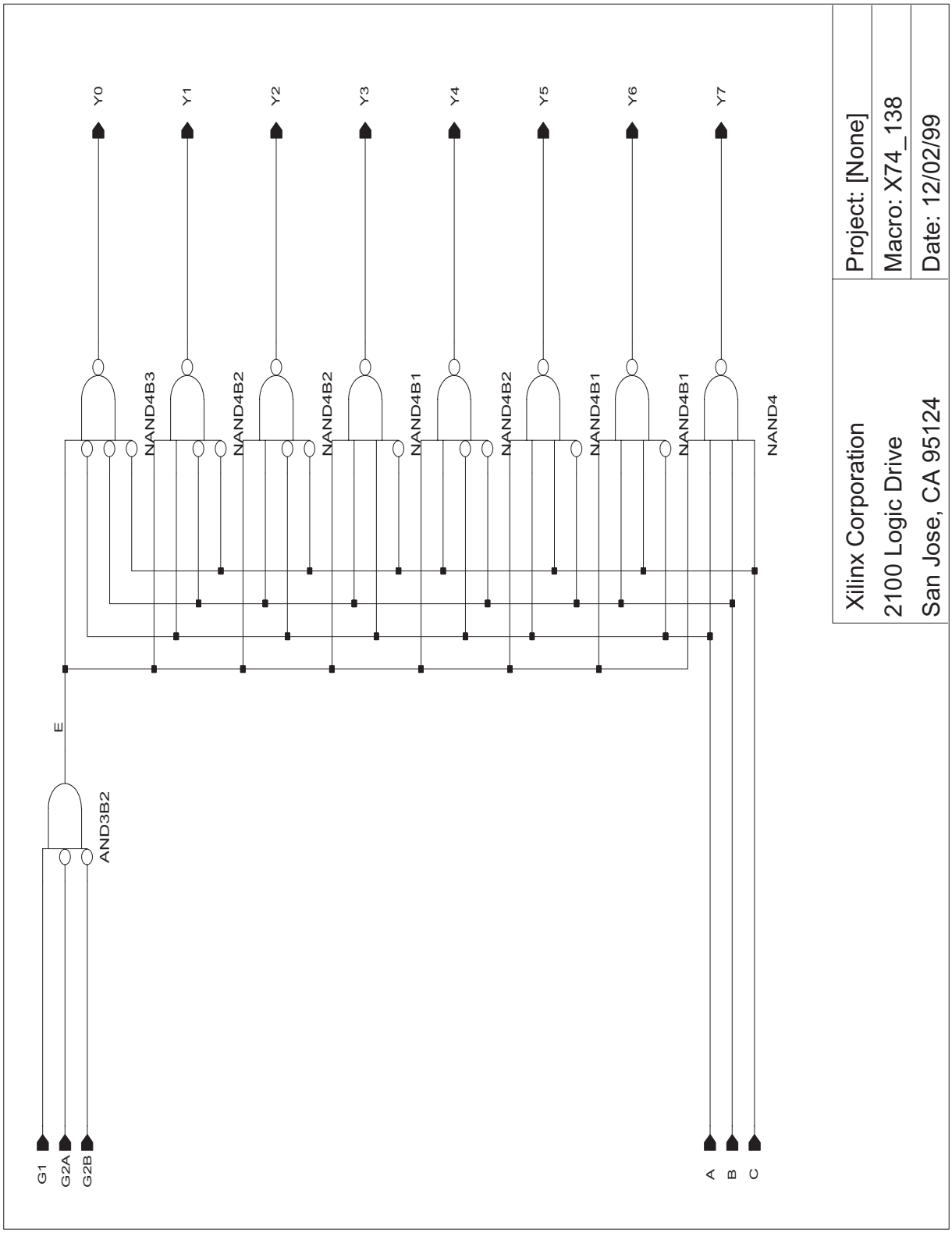


Figure K10, Detailed schematic of X74_138.

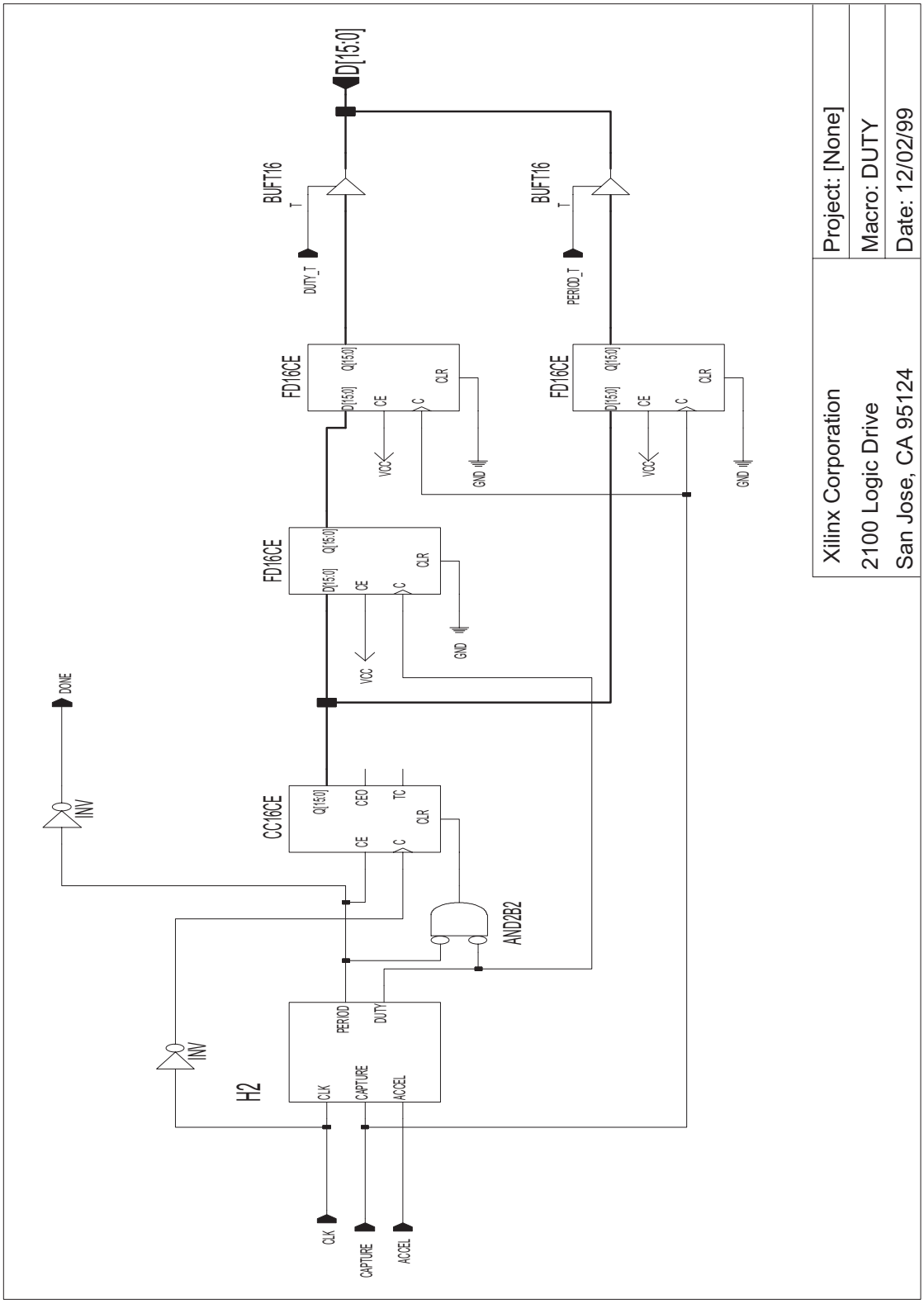


Figure K11, Detailed schematic of Duty.

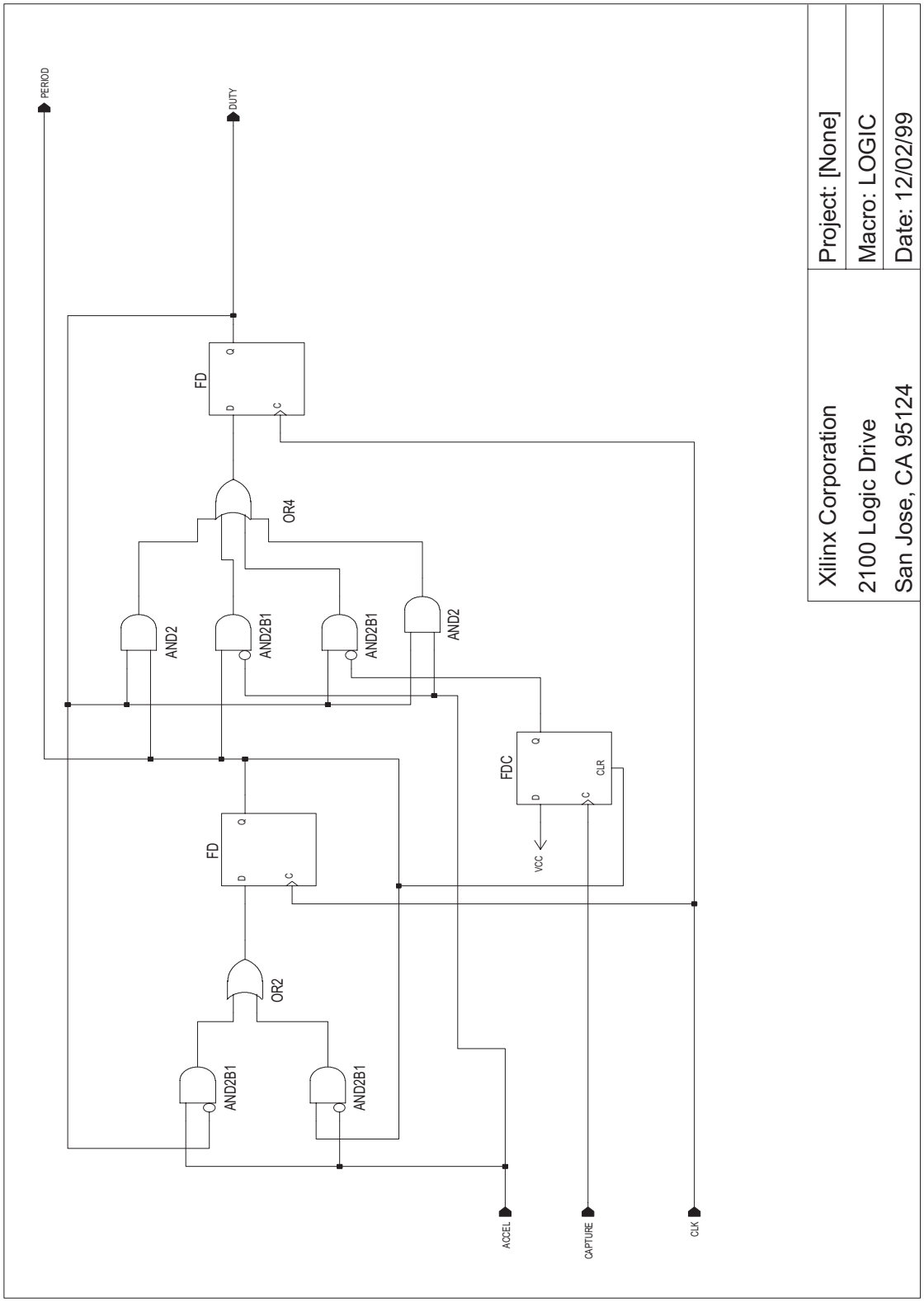


Figure K12, Detailed schematic of H2.

Appendix L: Data Acquisition Software.

Program One: This is the c program that was used in the micro-controller which interfaced between that FPGA to the laptop, to sent the data from the FPGA circuit-board to the laptop.

```
//-----------------------------------------------------------------------------
//         File:               target.c for use with accelerometers
//
//-----------------------------------------------------------------------------
#pragma NOIV                                        // Do not generate interrupt vectors
#define ALLOCATE_EXTERN  1     // This is needed to enable external memory acess of 8051
#include "ezusb.h"
#include "ezregs.h"

# define READ8              0
# define READ16             1
# define READ24             2
# define READ32             3
# define READ64             4
# define WRITE8             5
# define WRITE16                    6
# define WRITE24                    7
# define WRITE32                    8
# define WRITE64                    9
# define FPGA_CLEAR     10
# define FPGA_POST                  11
# define FPGA_STATUS    12
#define  READBLOCK      13

#define DELAY_COUNT   0x9248              // Delay for 1 sec
#define CLK                 0x20
#define DIN         0x08
#define PROGRAM           0x02
//pin 44 on 8051 is DONE which is PB0

//-----------------------------------------------------------------------------
// Random Macros
//-----------------------------------------------------------------------------
#define   min(a,b) (((a)<(b))?(a):(b))
#define   max(a,b) (((a)>(b))?(a):(b))

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------
```

```
volatile BOOL     GotSUD;
volatile BOOL     Sleep;
BOOL                    Rwuen;
BOOL                    Selfpwr;
BYTE                    Configuration;          // Current configuration
BYTE                    AlternateSetting;  // Alternate settings
        // Sleep mode enable flag

//-----------------------------------------------------------------------------
// Prototypes
//-----------------------------------------------------------------------------
void SetupCommand(void);
void TD_Init(void);
void TD_Poll(void);
BOOL DR_GetDescriptor(void);
BOOL DR_SetConfiguration(void);
BOOL DR_GetConfiguration(void);
BOOL DR_SetInterface(void);
BOOL DR_GetInterface(void);
BOOL DR_GetStatus(void);
BOOL DR_ClearFeature(void);
BOOL DR_SetFeature(void);
BOOL DR_VendorCmnd(void);

struct addressstruct
{
        unsigned char address[64];
}xdata mem _at_ 0x8300;


//-----------------------------------------------------------------------------
// Code
//-----------------------------------------------------------------------------

// Task dispatcher
void main(void)
{
        DWORD           j=0;

//mem_loop();
        // Initialize Global States
        Sleep = FALSE;                                  // Disable sleep mode
        Rwuen = FALSE;                                  // Disable remote wakeup
        Selfpwr = FALSE;                                // Disable self powered
        GotSUD = FALSE;                                         // Clear "Got setup data" flag

        // Initialize user device
        TD_Init();

        EZUSB_IRQ_ENABLE();                             // Enable USB interrupt (INT2)
        //EZUSB_ENABLE_RSMIRQ();                                // Wake-up interrupt

        USBBAV = USBBAV | 1 & ~bmBREAK;       // Disable breakpoints and autovectoring
        USBIEN |= bmSUDAV | bmSUTOK | bmSUSP | bmURES;      // Enable selected interrupts
        EA = 1;                                         // Enable 8051 interrupts

        // This loop waits until we receive a setup packet from the host.
        // NOTE: The device will continue to renumerate until it receives a setup
        // packet.  This fixes a microsoft USB bug that loses disconnect/reconnect
        // events during initial USB device driver configuration dialog box.
```

```
        // B2 Load:  This code is not needed for B2 load, only for renumeration.
        #ifndef NO_RENUM
                while(!GotSUD)
                {
                        if(!GotSUD)
                                EZUSB_Discon(TRUE);   // renumerate until setup received
                        for(j=0;(j<DELAY_COUNT) && (!GotSUD);++j);
                }
        #endif

  // Task Dispatcher
        while(TRUE)                                     // Main Loop
        {
                if(GotSUD)                              // Wait for SUDAV
                {
                        SetupCommand();                         // Implement setup command
                        GotSUD = FALSE;                         // Clear SUDAV flag
                }
                TD_Poll();
        }
}

// Device request parser
void SetupCommand(void)
{
        void    *dscr_ptr;
        DWORD           i;

        switch(SETUPDAT[1])
        {
                case SC_GET_DESCRIPTOR:                                         //
*** Get Descriptor
                        if(DR_GetDescriptor())
                                switch(SETUPDAT[3])
                                {
                                        case GD_DEVICE:                         //
Device
                                                SUDPTRH = MSB(&DeviceDscr);
                                                SUDPTRL = LSB(&DeviceDscr);
                                                break;
                                        case GD_CONFIGURATION:                  //
Configuration
                                                if(dscr_ptr = (void
*)EZUSB_GetConfigDscr(SETUPDAT[2]))
                                                {
                                                        SUDPTRH = MSB(dscr_ptr);
                                                        SUDPTRL = LSB(dscr_ptr);
                                                }
                                                else
                                                        EZUSB_STALL_EP0();     // Stall End
Point 0
                                                break;
                                        case GD_STRING:                         //
String
                                                if(dscr_ptr = (void
*)EZUSB_GetStringDscr(SETUPDAT[2]))
                                                {
                                                        // Workaround for rev D errata number
8
```

```
                                                // If you're certain that you will never
run on rev D,
                                                // you can just do this:
                                                // SUDPTRH = MSB(dscr_ptr);
                                                // SUDPTRL = LSB(dscr_ptr);
                                                STRINGDSCR *sdp;
                                                BYTE len;

                                                sdp = dscr_ptr;

                                                len = sdp->length;
                                                if (len > SETUPDAT[6])
                                                        len = SETUPDAT[6]; //limit to
the requested length

                                                while (len)
                                                {
                                                        for(i=0; i<min(len,64); i++)
                                                                *(IN0BUF+i) =
*((BYTE xdata *)sdp+i);

                                                        //set length and arm Endpoint

        EZUSB_SET_EP_BYTES(IN0BUF_ID,min(len,64));
                                                        len -= min(len,64);

                                                        // Wait for it to go out (Rev C
and above)
                                                        while(EP0CS & 0x04)
                                                                ;
                                                }

                                                // Arm a 0 length packet just in case.
There was some reflector traffic about
                                                // Apple hosts asking for too much data.
This will keep them happy and will
                                                // not hurt valid hosts because the next
SETUP will clear this.

        EZUSB_SET_EP_BYTES(IN0BUF_ID,0);
                                                // Clear the HS-nak bit
                                                EP0CS = bmHS;
                                        }
                                        else
                                                EZUSB_STALL_EP0();      // Stall End
Point 0
                                        break;
                                default:                                // Invalid request
                                        EZUSB_STALL_EP0();              // Stall End
Point 0
                                }
                        break;
                case SC_GET_INTERFACE:                                          //
*** Get Interface
                        DR_GetInterface();
                        break;
                case SC_SET_INTERFACE:                                          //
*** Set Interface
                        DR_SetInterface();
                        break;
```

```
                case SC_SET_CONFIGURATION:                          // *** Set
Configuration
                        DR_SetConfiguration();
                        break;
                case SC_GET_CONFIGURATION:                          // *** Get
Configuration
                        DR_GetConfiguration();
                        break;
                case SC_GET_STATUS:                                 // *** Get
Status
                        if(DR_GetStatus())
                                switch(SETUPDAT[0])
                                {
                                        case GS_DEVICE:                     //
Device
                                                IN0BUF[0] = ((BYTE)Rwuen << 1) |
(BYTE)Selfpwr;
                                                IN0BUF[1] = 0;
                                                EZUSB_SET_EP_BYTES(IN0BUF_ID,2);
                                                break;
                                        case GS_INTERFACE:          // Interface
                                                IN0BUF[0] = 0;
                                                IN0BUF[1] = 0;
                                                EZUSB_SET_EP_BYTES(IN0BUF_ID,2);
                                                break;
                                        case GS_ENDPOINT:           // End Point
                                                IN0BUF[0] =
EPIO[EPID(SETUPDAT[4])].cntrl & ~bmEPBUSY;
                                                IN0BUF[1] = 0;
                                                EZUSB_SET_EP_BYTES(IN0BUF_ID,2);
                                                break;
                                        default:                    // Invalid Command
                                                EZUSB_STALL_EP0();          // Stall End
Point 0
                                }
                        break;
                default:                                            // *** Invalid
Command
                        if(DR_VendorCmnd())
                                EZUSB_STALL_EP0();                  // Stall End
Point 0
        }

        // Acknowledge handshake phase of device request
        // Required for rev C does not effect rev B
        EP0CS |= bmBIT1;
}


void TD_Init(void)                          // Called once at startup
{
        //set up ports
        OEB=PROGRAM;        //Bit 1 for PROGRAM line
        OEC=CLK|DIN;        //Bits 3 and 5 for DIN and CCLK
        //Intial value
        OUTB=PROGRAM;
        OUTC=!CLK|!DIN;
        //Now set up ports for external memory access
        // Set the ports for alternate function -- rd/wr
        PORTCCFG |= 0xc0;       //RD WR
```

```
        PORTACFG |= 0x0C;   //CS OE

        // Enable endpoints
        IN07VAL = bmEP2+bmEP4+bmEP6;                    // Validate all EP's
        OUT07VAL = bmEP2+bmEP4+bmEP6;

        // Arm Endpoint 2 and 1 out to recieve data
        EPIO[OUT2BUF_ID].bytes = 0;
        EPIO[OUT4BUF_ID].bytes = 0;
        EPIO[OUT6BUF_ID].bytes = 0;

  // Enable the SOF and USB Reset interrupts
  USBIEN |= bmSOF + bmURES;
  // Enable the ISO IN endpoints
  INISOVAL = bmEP8;
  // Set up the addresses for the ISO buffers
  IN8ADDR = 0x00;
}


void TD_Poll(void)                              // Called repeatedly while the device is idle
{
        BYTE    count,count2,
                        i,num,bits,
                        command,address;

        ////////////////////////////////////////
  //FPGA
        ////////////////////////////////////////
        if(     !(EPIO[OUT6BUF_ID].cntrl & bmEPBUSY) )          // Is there something
in the OUT2BUF buffer,
                if( !(EPIO[IN6BUF_ID].cntrl & bmEPBUSY) )       // Is the IN2BUF available,
                {
                        count = EPIO[OUT6BUF_ID].bytes;         // Then loopback the
data
                        for(i=0;i<count;++i)
                        {
                                //echo data back to sender
                                IN6BUF[i] = OUT6BUF[i];
        //get recived data
                                num=OUT6BUF[i];
                                //divide up into individual bits
        for(count2=0;count2<8;count2++)
                                {
                                        bits=num&0x80;
                                        bits=bits>>7;
                                        if(bits) /* logic high */
                                        {
                                                /* din data */
                                                OUTC=!CLK|DIN;
                                                /* clear clk */
                                                OUTC=CLK|DIN;
                                        }
                                        else    /* logic low */
                                        {
                                                /*din data */
                                                OUTC=!CLK|!DIN;
                                                /* clear clk */
                                                OUTC=CLK|!DIN;
                                        }
```

```
                                        num=num<<1;
                                }//for()
                        }//for()
                        EPIO[OUT6BUF_ID].bytes = 0;
                        EPIO[IN6BUF_ID].bytes = count;
                }//if()

  ////////////////////////////////////////
        //Command
        ////////////////////////////////////////
        if(     !(EPIO[OUT4BUF_ID].cntrl & bmEPBUSY) )          // Is there something in the
OUT2BUF buffer,
        if( !(EPIO[IN4BUF_ID].cntrl & bmEPBUSY) )         // Is the IN2BUF available,
  {
                count = EPIO[OUT4BUF_ID].bytes;                         // Get the number of
bytes avaliable
     i=0;
                while(i<count)
                {
        command=OUT4BUF[0];             // Get Command
                address=OUT4BUF[1];             // Get Address

                switch(command)
                {
                        //Set program pulse low
                        case FPGA_CLEAR:
                                //Intial value
                                OUTB=PROGRAM;
                                OUTC=!CLK|!DIN;
                                //Pulse program line
                                OUTB=!PROGRAM;
                                OUTB=PROGRAM ;
                        break;

                        //Write to memory
                        case WRITE16:
        mem.address[address+1]=OUT4BUF[2];
                                mem.address[address]=OUT4BUF[3];
                        break;
                        //Read in one byte, address
                        //Read from memory
                        //Write back two bytes, datah,datal
                        case READ16:
        IN4BUF[2]=mem.address[address+1];
                                IN4BUF[3]=mem.address[address];

                        break;

                        default:
                                break;
                }//switch

                //Send back data
                IN4BUF[0]=command;
                IN4BUF[1]=address;
                i=i+4;

                }//while
                EPIO[OUT4BUF_ID].bytes = 0;
                EPIO[IN4BUF_ID].bytes = i;
```

```
        }//if
}


//-----------------------------------------------------------------------------
// Device Request hooks
//      The following hooks are called by the end point 0 device request parser.
//-----------------------------------------------------------------------------

BOOL DR_GetDescriptor(void)
{
        return(TRUE);
}

BOOL DR_SetConfiguration(void)  // Called when a Set Configuration command is received
{
        Configuration = SETUPDAT[2];
        return(TRUE);                           // Handled by user code
}

BOOL DR_GetConfiguration(void) // Called when a Get Configuration command is received
{
        IN0BUF[0] = Configuration;
        EZUSB_SET_EP_BYTES(IN0BUF_ID,1);
        return(TRUE);                           // Handled by user code
}

BOOL DR_SetInterface(void)              // Called when a Set Interface command is received
{
        AlternateSetting = SETUPDAT[2];
        return(TRUE);                           // Handled by user code
}

BOOL DR_GetInterface(void)              // Called when a Set Interface command is received
{
        IN0BUF[0] = AlternateSetting;
        EZUSB_SET_EP_BYTES(IN0BUF_ID,1);
        return(TRUE);                           // Handled by user code
}

BOOL DR_GetStatus(void)
{
        return(TRUE);
}


BOOL DR_VendorCmnd(void)
{
        return(TRUE);
}

//-----------------------------------------------------------------------------
// USB Interrupt Handlers
//      The following functions are called by the USB interrupt jump table.
//-----------------------------------------------------------------------------

// Setup Data Available Interrupt Handler
void ISR_Sudav(void) interrupt 0
{
        GotSUD = TRUE;                                  // Set flag
```

```
        EZUSB_IRQ_CLEAR();
        USBIRQ = bmSUDAV;                       // Clear SUDAV IRQ
}

// Setup Token Interrupt Handler
void ISR_Sutok(void) interrupt 0
{
        EPIO[OUT0BUF_ID].cntrl = 0;     // Clear endpoint 0 stall
        EZUSB_IRQ_CLEAR();
        USBIRQ = bmSUTOK;                       // Clear SUTOK IRQ
}

void delay(void)
{
        int j;
        for (j=0;j<10;j++);
}

void ISR_Sof(void) interrupt 0
{
        register BYTE framel = USBFRAMEL;
  register BYTE frameh = USBFRAMEH;
  register BYTE i;

        //Capture the signal, ie strobe capture line
  mem.address[1]=0;//HIGH THEN LOW
        mem.address[0]=0;//LOW
        delay();
        mem.address[1]=0;//HIGH
        mem.address[0]=1;//LOW
        delay();

        //First two bytes are frame number, byte order L then H
        IN8DATA = framel;
  IN8DATA = frameh;

        //Next 80 bytes are the 20 channels of data, 4 bytes per channel, byte order H then L
        //i<80;i=i+2)<<<<<<<<<<<<<<<<<<<<
  for (i=0;i<80;i=i+4)
        {
        //IN8DATA = mem.address[i+1];
                //IN8DATA = mem.address[i];
        IN8DATA = 0x14;
                IN8DATA = 0xa7;
        IN8DATA = 0x0a;
                IN8DATA = 0x9d;
        }

        EZUSB_IRQ_CLEAR();
        USBIRQ = bmSOF;                                 // Clear SOF IRQ
}


void ISR_Ures(void) interrupt 0
{
        EZUSB_IRQ_CLEAR();
        USBIRQ = bmURES;                        // Clear URES IRQ
}

void ISR_Susp(void) interrupt 0
```

```
{
        Sleep = TRUE;
        EZUSB_IRQ_CLEAR();
        USBIRQ = bmSUSP;
}
```

Program Two: This is the c program that was used in the laptop for receiving the data from the FPGA and saving the data to file.

```
/*++
File:               main.c for use with accelerometer
--*/

#include <windows.h>
#include <malloc.h>
#include <assert.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>
#include "resource.h"
#include "main.h"
#include <winioctl.h>
#include "ezusbsys.h"

#define ERROR_WRONG_PACKET_SIZE   0x01
#define ERROR_BULK_IN                   0x02
#define ERROR_BULK_OUT                  0x04
#define ERROR_OPEN_DRIVER         0x08
#define ERROR_OPEN_FILE           0x10
#define ERROR_DATA_MISMATCH             0x20
#define ERROR_FILE_0XFF                 0x40

#define COMMAND_FPGA_CLEAR              0x01
#define COMMAND_FPGA_POST         0x02
#define COMMAND_ADSETUP           0x04
#define COMMAND_MEMWRITE          0x08
#define COMMAND_MEMREAD           0x10
#define COMMAND_MEMWRITE2               0x20
#define COMMAND_MEMREAD2          0x40

#define COMMAND_ADGETDATA               0x01

#define PIPE_DATA_IN              0x04
#define PIPE_DATA_OUT                   0x05
#define PIPE_CONTROL_IN                 0x03
#define PIPE_CONTROL_OUT          0x02
#define PIPE_FPGA_IN              0x01
#define PIPE_FPGA_OUT                   0x00

#define READ8           0
#define READ16          1
#define READ24          2
#define READ32          3
#define READ64          4
```

```
#define WRITE8          5
#define WRITE16         6
#define WRITE24         7
#define WRITE32         8
#define WRITE64         9
#define FPGA_CLEAR      10
#define FPGA_POST               11
#define FPGA_STATUS     12
#define READBLOCK               13

#define  ADPRESCALE     0x02
#define  ADPERIOD               0x06
#define  ADCONTROL              0x00
#define FIFOSETUP               0x08

HWND            hOutputBox = NULL;
HINSTANCE hGInstance = NULL;
BOOL CALLBACK bMainDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam);
BOOL StopTest;
UCHAR usb_data[200];

void DataDisplay(LPVOID lpParameter);
UCHAR USBCommand(PCHAR device, UCHAR command, UCHAR address, PCHAR data);
UCHAR USB_FPGA_Download(PCHAR device, PCHAR filename);
void DisplayErrorMessage(ULONG error);
BOOL BufferHasErrors(PUCHAR buffer, PUCHAR pattern, int length);
char *build_time = __TIME__;
char *build_date = __DATE__;


/*******************************
*WinMain: Windows Entry point  *
*******************************/
int PASCAL WinMain(HINSTANCE hInstance,
            HINSTANCE hPrevInstance,
            LPSTR     lpCmdLine,
            int       nCmdShow)
{
  hGInstance=hInstance;

  if(DialogBox(hInstance,"MAIN_DIALOG",NULL,(DLGPROC)bMainDlgProc)==-1)
    MessageBox(NULL,"Unable to create root dialog!","DialogBox failure",MB_ICONSTOP);

  return 0;
}



void DataCaptureThread(LPVOID lpParameter)
{
  HWND hDlg = (HWND) lpParameter;
  HWND    hOutputBox              = NULL;
  HANDLE  hDevice                 = NULL;
  char    pcDriverName[MAX_DRIVER_NAME]   = "Ezusb-0";
  BOOLEAN bResult                 = FALSE;
  int     nBytes                  = 0;
  int     nItems                  = 0;
  ISO_TRANSFER_CONTROL                    IsoControl;
          PUSBD_ISO_PACKET_DESCRIPTOR     isoDesc;
```

```
ULONG               bytesToRead;
BOOL               Success;
PUCHAR              buffer = NULL;
ULONG              packetsRead;
ULONG              i;
PUCHAR              ptr;
ULONG              pass = 0;
UCHAR              count;
ULONG              framesRead = 0;
ULONG              framesMissed = 0;
char   pcDataFilename[40]   = "";
        PUCHAR          ptroutfile;
        ULONG           countoutfile;
        int    secondstolog=0;
        HANDLE hFile;
        DWORD  dwBytesWritten;
        char            buff[10];
        FILE            *outfile;

        // Get a handle to the output box
hOutputBox = GetDlgItem (hDlg, IDC_OUTPUT_BOX);
        //Get the filename from dialogue box
        GetDlgItemText (hDlg, IDC_DATA_FILENAME, pcDataFilename, MAX_DRIVER_NAME);

        //Open up file for writing
        //hFile = CreateFile(pcDataFilename,      // create MYFILE.TXT
//      GENERIC_WRITE,              // open for writing
//      0,                      // do not share
//      NULL,                   // no security
//      CREATE_ALWAYS,              // overwrite existing
//      FILE_ATTRIBUTE_NORMAL |     // normal file
//      FILE_FLAG_OVERLAPPED,       // asynchronous I/O
//      NULL);
                        // no attr. template

outfile = fopen(pcDataFilename,"wb");


       //Check to see if we opened the file
       //if (hFile == INVALID_HANDLE_VALUE)
       //{
               //Inform user and exit if file did not open
// SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Failed to open file");
       //      CloseHandle(hFile);
       //      return;
       //}

// Open the driver
if (bOpenDriver (&hDevice, pcDriverName) == FALSE)
        {
        //Inform user and exit if driver did not open
 SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Failed to Open Driver");
                CloseHandle (hDevice);
                CloseHandle(hFile);
 return;
}
                //Set up some parameters
        secondstolog=(GetDlgItemInt(hDlg,IDC_SECONDSTOLOG,&Success,FALSE))*1000;
IsoControl.PacketSize = 82;
IsoControl.PacketCount = 128;
```

```
  IsoControl.PipeNum = 7;
  IsoControl.BufferCount = 2;
  IsoControl.FramesPerBuffer = 8;

  bytesToRead = IsoControl.PacketCount * (IsoControl.PacketSize +
sizeof(USBD_ISO_PACKET_DESCRIPTOR));
  //Allocate memory for data transfer buffer
        buffer = (PUCHAR) malloc(bytesToRead);
  if (!buffer)
  {
    SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Alloc Failed");
    CloseHandle(hDevice);
    return;
  }

  while (!StopTest)
  {
                        //Read from USB port
    bResult = DeviceIoControl (hDevice,
            IOCTL_EZUSB_READ_ISO_BUFFER,
            &IsoControl,
            sizeof(ISO_TRANSFER_CONTROL),
            buffer,
            bytesToRead,
            &nBytes,
            NULL);

                        //Check for read fail
    if (bResult != TRUE)
    {
      SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Read ISO Buffer Failed");
      break;
    }
    ptr = buffer;
    packetsRead = nBytes / (IsoControl.PacketSize + sizeof(USBD_ISO_PACKET_DESCRIPTOR));
    isoDesc = (PUSBD_ISO_PACKET_DESCRIPTOR) (buffer + (packetsRead *
IsoControl.PacketSize));
    for (i=0;i<packetsRead;i++)
    {
      framesRead++;
                                //Update frames read dialogue
                                //Update grid display 3 time a second
      if (!(framesRead % 1000))
      {
        SetDlgItemInt(hDlg,IDC_FRAMES_READ,framesRead/1000,FALSE);
                                        DataDisplay(hDlg);
      }

                                //If time up then stop ISO transfer
                                if (!(framesRead % secondstolog))
      {
                                        StopTest=TRUE;
                                        // Open the driver
                                        if (bOpenDriver (&hDevice, pcDriverName) ==
FALSE)
                                        {
                                                SendMessage (hOutputBox,
LB_ADDSTRING, 0, (LPARAM)"Failed to Open Driver");
                                                hDevice = NULL;
                                        }
```

```
                                        //Send stop command
                                        bResult = DeviceIoControl (hDevice,
                IOCTL_EZUSB_STOP_ISO_STREAM,
                NULL,
                0,
                NULL,
                0,
                &nBytes,
                NULL);

                                        if (bResult != TRUE)
                                        {
                                                SendMessage (hOutputBox,
LB_ADDSTRING, 0, (LPARAM)"ISO stop command failed");
                                                CloseHandle (hDevice);
                                        }
                                        CloseHandle (hDevice);
                                }
                                ////////////////////////////////////////////////////////////////
                                //File write rotine
                                ptroutfile=ptr;
                                if(isoDesc[i].Length!=82)
                                {
                                        SendMessage (hOutputBox, LB_ADDSTRING, 0,
(LPARAM)"Wrong length recived");
                                }
                                else
                                {
                                        //Header
                                        buff[0]=(char)0xff;
                                        fwrite(buff,sizeof(char),1,outfile);
                                        fwrite(ptr,sizeof(char),isoDesc[i].Length,outfile);

                                        //
          WriteFile(hFile,buff,1,&dwBytesWritten, NULL);
                                        //          WriteFile(hFile, ptr,
isoDesc[i].Length,&dwBytesWritten, NULL);


          for(countoutfile=0;countoutfile<(isoDesc[i].Length);countoutfile++)
                                        {
                                                usb_data[countoutfile]=*ptroutfile;
                                                *ptroutfile++;
                                        }
                                        //Fotter
                                        buff[0]=0x00;
                                        fwrite( buff,sizeof(char),1,outfile );
                                }
                                //WriteFile(hFile,buff,1,&dwBytesWritten, NULL);
                                ////////////////////////////////////////////////////////////////
                                //Check for missed frames
        if (count != *ptr)
        {
          framesMissed++;
          SetDlgItemInt(hDlg,IDC_FRAMES_MISSED,framesMissed,FALSE);
          count = *ptr;
        }
        count++;
                                //Check for short packet
        if (isoDesc[i].Length != IsoControl.PacketSize)
```

```
      {
        framesMissed++;
        SetDlgItemInt(hDlg,IDC_FRAMES_MISSED,framesMissed,FALSE);
        SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Short Packet received");
      }
      ptr += IsoControl.PacketSize;
    }
  }
        CloseHandle (hDevice);
        //CloseHandle(hFile);
        fclose(outfile);
        free(buffer);
}//DataCaptureThread()


void DataDisplay(LPVOID lpParameter)
{
        HWND  hDlg                                                      = (HWND)
lpParameter;
        int             T1,T2,counter;
        float   acceleration;
        char    acceleration_s[10];

        //Calculate data
        for(counter=0;counter<20;counter++)
        {
                //calculate period and duty
                //Changed T2 and T1 arround
                T2=     ((usb_data[counter*4+2]<<8)|usb_data[counter*4+3]);
                T1=     ((usb_data[counter*4+4]<<8)|usb_data[counter*4+5]);
                acceleration=(((float)T1/(float)T2)-(float)0.5)/(float)0.125;
                //if((acceleration<0.001)|(acceleration>1000))
                //      acceleration=0;
                //Convert float into string
                sprintf(acceleration_s,"%2.2f",acceleration);
                //Display the answer on screen
                SetDlgItemText(hDlg,2000+counter,acceleration_s);
        }
}//DataDisplay()


/***********************************************
* Main Dialog proc                             *
***********************************************/

BOOL CALLBACK bMainDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
  HWND                  hOutputBox              = NULL;
  HANDLE                hDevice                 = NULL;
  char                  pcDriverName[MAX_DRIVER_NAME]   = "Ezusb-0";
  BOOLEAN       bResult                 = FALSE;
  int                   nBytes                  = 0;
  int                   nItems                  = 0;
  HFONT                 hFont                   = NULL;
  ULONG                 ulLength                = 0;
  char                  tempbuff[256];
  UCHAR                 errorcode;
```

```
  char                    pcFPGAFilename[40]   = "";
        int                       flag=0;

  // Get a handle to the output box
  hOutputBox = GetDlgItem (hDlg, IDC_OUTPUT_BOX);

  MAINTAIN_OUTPUT_BOX (hOutputBox, nItems);
  switch(message)
  {
  case WM_LBUTTONDOWN:
    hOutputBox = GetDlgItem (hDlg, IDC_OUTPUT_BOX);

    MAINTAIN_OUTPUT_BOX (hOutputBox, nItems);

    SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"you clicked");
    break;

  case WM_INITDIALOG:
    // Get a handle to the output box
    hOutputBox = GetDlgItem (hDlg, IDC_OUTPUT_BOX);

    // Setup the std system font
    hFont = GetStockObject(SYSTEM_FONT);
    SendMessage (hOutputBox, WM_SETFONT, (WPARAM)hFont, MAKELPARAM(TRUE,0));
    sprintf(tempbuff,"Build  %s  %s",build_time,build_date);
    SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)tempbuff);

    // Setup the default symbolic name for the device driver
    SetDlgItemText (hDlg, IDC_FPGA_FILENAME, "C:\\usb\\USBACCEL.BIT");
                    SetDlgItemText (hDlg, IDC_DATA_FILENAME,
"C:\\usb\\DATALOG.DAT");

                    SetDlgItemInt(hDlg,IDC_SECONDSTOLOG,20,FALSE);

    break; /*end WM_INITDIALOG case*/

  case WM_COMMAND:
    switch(LOWORD(wParam))
    {

    case IDC_QUIT:
      EndDialog(hDlg,0);
      break;

    case IDC_CLEAR:
      hOutputBox = GetDlgItem (hDlg, IDC_OUTPUT_BOX);
      SendMessage (hOutputBox, LB_RESETCONTENT, 0, 0);
    break;
                    ////////////////////////////////////
                    case    IDC_FPGADOWNLOAD:
                            //Download data to FPGA
                            GetDlgItemText (hDlg, IDC_FPGA_FILENAME,
pcFPGAFilename, MAX_DRIVER_NAME);
                            errorcode=USB_FPGA_Download("Ezusb-0",pcFPGAFilename);
                            DisplayErrorMessage(errorcode);
                            SendMessage (hOutputBox, LB_ADDSTRING, 0,
(LPARAM)"FPGA programmed");
                    break;

    ////////////////////////////////////
```

```
case IDC_START:
{
  ISO_TRANSFER_CONTROL IsoControl;
  ULONG                pipenum;
  ULONG                count=0;

  // Open the driver
  if (bOpenDriver (&hDevice, pcDriverName) == FALSE)
                                    {
   SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Failed to Open Driver");
   hDevice = NULL;
   break;
  }

  IsoControl.PacketSize = 82;


  // reset the pipe
  pipenum = 7;

  bResult = DeviceIoControl (hDevice,
            IOCTL_Ezusb_RESETPIPE,
            &pipenum,
            sizeof(ULONG),
            NULL,
            0,
            &nBytes,
            NULL);

  if (bResult != TRUE)
  {
    SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Pipe Reset Failed");
    CloseHandle (hDevice);
    break;
  }
  // perform the ISO transfer
  IsoControl.PacketCount                = 128;
  IsoControl.PipeNum                            = 7;
  IsoControl.BufferCount                = 2;
  IsoControl.FramesPerBuffer = 8;

  bResult = DeviceIoControl (hDevice,
              IOCTL_EZUSB_START_ISO_STREAM,
              &IsoControl,
              sizeof(ISO_TRANSFER_CONTROL),
              NULL,
              0,
              &nBytes,
              NULL);

  if (bResult != TRUE)
  {
    SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"ISO Transfer Failed");
    CloseHandle (hDevice);
    break;
  }

  CloseHandle (hDevice);

                                StopTest = FALSE;
```

```
      _beginthread(DataCaptureThread,0,hDlg);
    }
    break;

    case IDC_STOP:
    {
                                          StopTest = TRUE;
      // Open the driver
      if (bOpenDriver (&hDevice, pcDriverName) == FALSE)
                                        {
       SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"Failed to Open Driver");
       hDevice = NULL;
       break;
      }

      bResult = DeviceIoControl (hDevice,
                IOCTL_EZUSB_STOP_ISO_STREAM,
                NULL,
                0,
                NULL,
                0,
                &nBytes,
                NULL);

      if (bResult != TRUE)
      {
        SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"ISO Transfer Failed");
        CloseHandle (hDevice);
        break;
      }

      CloseHandle (hDevice);
    }
    break;


    } /*end switch wParam*/

    break;


  } /*end switch message*/

  return FALSE;

} /*end MainDlgProc*/




UCHAR USBCommand(PCHAR device, UCHAR command, UCHAR address, PCHAR data)
{
          BULK_TRANSFER_CONTROL bulkControl;
          HANDLE  hDevice                       = NULL;
          BOOLEAN bResult                       = FALSE;
          UCHAR errorCode                 = 0x00;
          WORD   PacketSize;
          UCHAR dataBuffer[18];
  ULONG           nBytes;
          PCHAR  datafirst;
```

```
        int                     count;

        //Record position of first item in pointer list
        datafirst=data;

// Open the driver
        bResult=bOpenDriver(&hDevice, device);
if (bResult != TRUE)
        {
                errorCode=errorCode|ERROR_OPEN_DRIVER;
                return(errorCode);
        }
        else
        {
        //Fill out buffer and set packet size
        PacketSize=4;
        dataBuffer[0]=command;
        dataBuffer[1]=address;
        switch(command)
        {
                case FPGA_CLEAR:
                        break;
                case READ8:
                        break;
                case WRITE8:
                        dataBuffer[2]=*data;
                        break;
                case READ16:
                        break;
                case WRITE16:
                        dataBuffer[2]=*data++;
                        dataBuffer[3]=*data;
                        break;
                default:
                        break;
        }//switch()
        //Set pipe number, command out
        bulkControl.pipeNum = 4;
        // Perform the BULK OUT
bResult = DeviceIoControl (hDevice,
                IOCTL_EZUSB_BULK_WRITE,
                 &bulkControl,
                 sizeof(BULK_TRANSFER_CONTROL),
                 &dataBuffer[0],
                 PacketSize,
                 &nBytes,
                 NULL);
        if (bResult != TRUE)
                errorCode=errorCode|ERROR_BULK_OUT;

                //Set pipe number, command in
                bulkControl.pipeNum = 3;
                // Perform the BULK IN
                bResult = DeviceIoControl (hDevice,
                 IOCTL_EZUSB_BULK_READ,
                 &bulkControl,
                 sizeof(BULK_TRANSFER_CONTROL),
                 &dataBuffer[0],
                 PacketSize,
                 &nBytes,
```

```
                NULL);
                //Check for error conditions
                if (bResult != TRUE)
                        errorCode=errorCode|ERROR_BULK_IN;
                if (nBytes != PacketSize)
                        errorCode=errorCode|ERROR_WRONG_PACKET_SIZE;
                //Transfer recived data to correct location
                data=datafirst;
                for(count=2;count<PacketSize;count++)
                {
                        *data++=dataBuffer[count];
                }
        CloseHandle(hDevice);
        }//else
        return(errorCode);
}//USBCommand()



//This function displays an error message
void DisplayErrorMessage(ULONG error)
{

        switch(error)
        {
                case    ERROR_WRONG_PACKET_SIZE:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0,
(LPARAM)"ERROR:Wrong packet size recived");
                        break;
                case    ERROR_BULK_IN:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"ERROR:Bulk
in failed");
                        break;
                case    ERROR_BULK_OUT:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"ERROR:Bulk
out failed");
                        break;
                case    ERROR_OPEN_DRIVER:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0,
(LPARAM)"ERROR:Could not open driver");
                        break;
                case    ERROR_OPEN_FILE:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0,
(LPARAM)"ERROR:Could not open file");
                        break;
                case    ERROR_DATA_MISMATCH:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)"ERROR:Data
mismatch");
                        break;
                case    ERROR_FILE_0XFF:
                        SendMessage (hOutputBox, LB_ADDSTRING, 0,
(LPARAM)"ERROR:Could not find 0xff");
                        break;


        }
}//DisplayErrorMessage()


UCHAR USB_FPGA_Download(PCHAR device, PCHAR filename)
```

```
{
        BULK_TRANSFER_CONTROL bulkControl;
        HANDLE          hDevice                 = NULL;
        BOOLEAN         bResult                 = FALSE;
        int             nBytes                  = 0;
        FILE    *infile;
        int             num,counter,sentpackets=0;
        UCHAR outBuffer[64], inBuffer[64];
        WORD   PacketSize,counter2;
        WORD   fileTransferSize=0;
        UCHAR errorCode                 = 0x00;
        div_t   packetsToSend;
        fpos_t  fileposition;
        int             fecounter;
        UCHAR data[4];

        //Reset the FPGA
        USBCommand("Ezusb-0",FPGA_CLEAR,0x00,data);
        // open file
        if(!(infile=fopen(filename,"rb")))
        {
                errorCode=errorCode|ERROR_OPEN_FILE;
                return(errorCode);
        }
        // find the first bits to send to FPGA
        num=0x00;
        fecounter=0;
        while(!(num==0xff))
        {
                fgetpos(infile,&fileposition);
                num=fgetc(infile);
                fecounter++;
                if(fecounter>1000)
                {
                        errorCode=errorCode|ERROR_FILE_0XFF;
                        return(errorCode);
                }
        }
        //Calculate the total size of file to send
        fileTransferSize=0;
        do
        {
                fileTransferSize++;
                num=fgetc(infile);
        }while(!feof(infile));
        //Set packet size
        PacketSize = 0x30;
        //Claculate the number of packets to send
        packetsToSend=div((fileTransferSize),PacketSize);
        //Set file position to 0xff,ready to send data
        fsetpos(infile,&fileposition);
        // Open the driver
        bResult=bOpenDriver(&hDevice, "Ezusb-0");

    if (bResult != TRUE)
        {
                errorCode=errorCode|ERROR_OPEN_DRIVER;
                return(errorCode);
        }
        //Loop to send all packets of size 0x30
```

```
        for(counter2=0;counter2<packetsToSend.quot;counter2++)
        {
                //Read in data and fill up send buffer
                for(counter=0;counter<PacketSize;counter++)
                {
                        outBuffer[counter]=fgetc(infile);
                }

    bulkControl.pipeNum = 6;

    // Perform the BULK OUT
    bResult = DeviceIoControl (hDevice,
                IOCTL_EZUSB_BULK_WRITE,
                 &bulkControl,
                 sizeof(BULK_TRANSFER_CONTROL),
                 &outBuffer[0],
                 PacketSize,
                 &nBytes,
                 NULL);

                if (bResult != TRUE)
                        errorCode=errorCode|ERROR_BULK_OUT;

    // Read from each IN pipe and verify the data
    bulkControl.pipeNum = 5;
    // Perform the BULK IN
    bResult = DeviceIoControl (hDevice,
                 IOCTL_EZUSB_BULK_READ,
                 &bulkControl,
                 sizeof(BULK_TRANSFER_CONTROL),
                 &inBuffer[0],
                 PacketSize,
                 &nBytes,
                 NULL);

    if (bResult != TRUE)
                        errorCode=errorCode|ERROR_BULK_IN;

                if (nBytes != PacketSize)
                        errorCode=errorCode|ERROR_WRONG_PACKET_SIZE;

    if (BufferHasErrors(&inBuffer[0], &outBuffer[0], PacketSize))
                        errorCode=errorCode|ERROR_DATA_MISMATCH;
}//for
        ////////////////////////////////////
        //Send the remaining patial packet
        //Read in data and fill up send buffer
        PacketSize  = packetsToSend.rem;
        for(counter=0;counter<packetsToSend.rem;counter++)
        {
                num=fgetc(infile);
                outBuffer[counter]=num;
        }

bulkControl.pipeNum = 6;
// Perform the BULK OUT
bResult = DeviceIoControl (hDevice,
                 IOCTL_EZUSB_BULK_WRITE,
                 &bulkControl,
                 sizeof(BULK_TRANSFER_CONTROL),
```

```
                    &outBuffer[0],
                    PacketSize,
                    &nBytes,
                    NULL);
        //Check for errors
        if (bResult != TRUE)
                errorCode=errorCode|ERROR_BULK_OUT;

  // Read from each IN pipe and verify the datA
  bulkControl.pipeNum = 5;
  // Perform the BULK OUT
  bResult = DeviceIoControl (hDevice,
                    IOCTL_EZUSB_BULK_READ,
                    &bulkControl,
                    sizeof(BULK_TRANSFER_CONTROL),
                    &inBuffer[0],
                    PacketSize,
                    &nBytes,
                    NULL);
        //Check for errors
  if (bResult != TRUE)
                    errorCode=errorCode|ERROR_BULK_IN;
        if (nBytes != PacketSize)
                    errorCode=errorCode|ERROR_WRONG_PACKET_SIZE;
  if (BufferHasErrors(&inBuffer[0], &outBuffer[0], PacketSize))
                    errorCode=errorCode|ERROR_DATA_MISMATCH;
        //Tidy up by closing used handlers
        fclose(infile);
        CloseHandle(hDevice);
        return(errorCode);
}



#define BYTES_PER_LINE 0x10

void
DumpBuffer(PVOID pvBuffer, ULONG length, HWND hOutputBox)
{
  int               nItems    = 0;
  char              temp[64]  = "";
  char              temp2[64]  = "";
  ULONG             i;
  ULONG   j;
  PUCHAR            ptr;

  MAINTAIN_OUTPUT_BOX (hOutputBox, nItems);

  ptr = (PUCHAR) pvBuffer;

  for (i = 0; i < ((length + BYTES_PER_LINE - 1) / BYTES_PER_LINE); i++)
  {
    wsprintf(temp,"%04X ",(i*BYTES_PER_LINE));
    for (j = 0; j < BYTES_PER_LINE; j++)
    {
      if (((i * BYTES_PER_LINE) + j) < length)
      {
        wsprintf(temp2,"%02X ",*ptr++);
        strcat(temp,temp2);
```

```
      }
    }
    SendMessage (hOutputBox, LB_ADDSTRING, 0, (LPARAM)temp);
  }
}

/***********************************************
* bOpenDriver proc                             *
*                                              *
* Purpose:                                     *
*     Opens the device driver using symbolic   *
*     name provided                            *
*                                              *
* Input:                                       *
*     phDeviceHandle:                          *
*        Pointer to Device Driver handle where *
*        the file handle is placed.            *
*     devname:                                 *
*        Null terminated string containing the *
*        device name                           *
*                                              *
* Return Value:                                *
*     Boolean that indicates if the driver was *
*     successfully opened or not.              *
*                                              *
***********************************************/

BOOLEAN
bOpenDriver (HANDLE * phDeviceHandle, PCHAR devname)
{
  char completeDeviceName[64] = "";
  char pcMsg[64] = "";

  strcat (completeDeviceName,
    "\\\\.\\"
    );

  strcat (completeDeviceName,
                devname
       );

  *phDeviceHandle = CreateFile(   completeDeviceName,
    GENERIC_WRITE,
    FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    0,
    NULL);

  if (*phDeviceHandle == INVALID_HANDLE_VALUE) {
    return (FALSE);
  } else {
    return (TRUE);
  } /*else*/


}//OpenDevice
```

```
BOOL BufferHasErrors(PUCHAR bufferin, PUCHAR bufferout, int length)
{
  int i;

  for (i = 0; i < length; i++)
  {
    if (bufferin[i] != bufferout[i])
    {
      return TRUE;
    }

  }
  return FALSE;
}
```

Appendix M: Data Analysis Software

Directory of files:

full_data_aquthesis.m:

This is the main calling program of the post processing and analysis of the experimental results. It calls all of the functions that follow.

```
disp('The sampling frequency is (Hz):'); %Display what is in the
quotations.
s_f=1000; %Calculates the sampling frequency.
disp(s_f);
disp('The sampling period is (s):');
s_p=1/s_f;  %Calculates the sampling period.
disp(s_p);
fn=1;
old_time=clock%this is so that you can see that the program is running
tc=30%this is the time count for the update intervals
program_run=input('Before running this program, have you REMEMBERED to
change \n the file name that this program will save the data to so that
\n previously saved data is not over written, press 1 for \n ''yes that
you have'' and ''0'' if not: \n');
if program_run==1
   fft_decision=input(' \n \n Do you want the Fourier Series analysis
done, 1=Yes, 0=No: \n \n >:');
   %The following command calls the following function, namely
INPUT_FILES

[name,fr,test_cell,r,c,length_vec,NODOF,test_t,test_cell_name,fn,total,
rotcal]=full_input_files(fn,s_p);
   fn=fn-1;
   k=total;%calibration file required for the calibration of the
accelerometer

   [vec_cal_cell,AOK]=full_cal_files2(k);

[R,t_cal_rCELL,testcal,dir_cal_cell,test_cell_3D,stat_cell_3D,stat_cell
_2D,cal_cel]=full_G_calibration(AOK,vec_cal_cell,test_cell,length_vec,k
,NODOF,rotcal);

[acel,vel,old_time]=full_velocity(tc,old_time,t_cal_rCELL,k,length_vec,
test_t);
   [dis,old_time]=full_displacment(tc,old_time,vel,k,length_vec,test_t)
;
```

```
   if fft_decision==1
      %this is to calculate the fourier series

   [F_uc,A0,An,Bn,N,x,T,old_time]=fft_to_series(tc,old_time,k,test_t,t_
cal_rCELL);

      %this is for the first integration of the fourier series
      [F_uc_i1,old_time]=fft_integral(tc,old_time,k,A0,An,Bn,N,x,T);

      %this is for the second integration of the fourier series
      [F_uc_i2,old_time]=fft_inter_2nd(tc,old_time,k,A0,An,Bn,N,x,T);
   end
   %this calculates and graphs the power density spectrum

[qa_power,qa_freq,qd_power,qd_freq,old_time]=full_Quintons_PDS(tc,old_t
ime,k,acel,s_f,dis);
   %this is for the (filtered) jerk (rate of change of acceleration)

[filter_rocoa,old_time]=full_jerk(tc,old_time,k,acel,s_p,length_vec,tes
t_t,test_cell_name);
   if fft_decision==1

   [old_time,a_RMS,v_RMS,d_RMS,a_RMS_f_series,v_RMS_f_series,d_RMS_f_se
ries]=RMS_norm(tc,old_time,t_cal_rCELL,vel,dis,k,length_vec,test_t,s_p,
F_uc,F_uc_i1,F_uc_i2);
   else

   [old_time,a_RMS,v_RMS,d_RMS]=full_RMS_norm(tc,old_time,acel,vel,dis,
k,length_vec,test_t,s_p);
   end
   if fft_decision==1

   [fig_no]=full_figure_functions(s_f,s_p,name,fr,test_cell,r,c,length_
vec,test_t,test_cell_name,fn,total,k,vec_cal_cell,R,t_cal_rCELL,vel,dis
,qa_power,qa_freq,qd_power,qd_freq,q_peeks,poss_num_of_peeks,filter_roc
oa,a_RMS,v_RMS,d_RMS,F_uc,A0,An,Bn,N,x,T,F_uc_i1,F_uc_i2,a_RMS_f_series
,v_RMS_f_series,d_RMS_f_series);
   else

   [fig_no]=full_figure_functions(s_f,s_p,name,fr,test_cell,r,c,length_
vec,test_t,test_cell_name,fn,total,k,vec_cal_cell,R,t_cal_rCELL,acel,ve
l,dis,qa_power,qa_freq,qd_power,qd_freq,q_peeks,poss_num_of_peeks,filte
r_rocoa,a_RMS,v_RMS,d_RMS);
   end
else
   err_warn=sprintf('You MUST change the name of the save file before
running this program, \n so that data is not over written from a
previous running of this program. \n\n This can be done very easily by
going into this main \n computer program and where to change the \n
file name is also well indicated!!!.');
   error(err_warn);

end
```

full_input_filesthesis.m:

This function loads all of the files that were created from the experimental trials and stores them in a cell array, so that they can be calibrated later.

```
function
[name,fr,test_cell,r,c,length_vec,NODOF,test_t,test_cell_name,fn,total,
rotcal]=full_input_files(fn,s_p)
total=0;
while fn
   textnames=input('\n \n Enter in the name of file (*.txt) containing
test run names. \n >:','s');
   %R = INPUT('What is your name','s') gives the prompt in the text
   %string and waits for character string input.
   if isempty(textnames),; %ISEMPTY True for empty matrix.
                           %ISEMPTY(X) returns 1 if X is an empty array
and 0 otherwise.
      break;               %BREAK Terminate execution of WHILE or FOR
loop.
   end;  %hence once all textfiles are entered hitting the return
button when it asks you
         %for the next file will exit the loop.
   [name]=textread(textnames,'%s');
   [fr,fc]=size(name); %fr is the number file data file the program has
to analyse
   test_tot=0;
   for fk=1:fr;
      test_no=name(fk)%this assigns every data file a test number.
      [accel,OK]=hex_accel(test_no{1});
      %evalfile_str=sprintf('test=load(''%s'');',test_no{1}); %SPRINTF
Write formatted data to string.
      %eval(evalfile_str); %EVAL Execute string with MATLAB expression.
                           %EVAL(s), where s is a string, causes MATLAB
to execute
                           %the string as an expression or statement.
      if OK==1
         test_tot=test_tot+1
         test=accel;
         test_cell(1,test_tot+total)={test};%puts all the data for each
data file into separate cells of a cell array.
         [r,c]=size(test); %this calculates the matrix size of each
data file (test) in terms of the number of row and columns.
         length_vec(1,test_tot+total)=r;  %creates a vector contain
information of the size of each test.
         NODOF(1,test_tot+total)=c-1;  %%creates a vector contain
information of the Number Of Degrees Of Freedom (NODOF) of each test.
         t=0:s_p: length_vec(1,test_tot+total)*s_p;   %calculate a
vector of the time step for each test.
         test_t(1,test_tot+total)={t(1:length_vec(1,test_tot+total))};  %stores each ti
         %the next three line are the same as above but is for storing
the name of each data file that is associated
         %with each test for identification of graphs and figures.
```

```
         evalfile_str1 =
sprintf('test%d=load(''%s'');',test_tot+total,test_no{1});
         test_cell_name(1,test_tot+total)={evalfile_str1};
      else
         warning(test_no{1})
         disp('was not loaded the file is corrupt')
      end
   end
   total = test_tot+total; %this is the running title of how many test
are enter to be analyised.
   fn=fn+1;
end
```

hex_accelthesis.m:

This function is called by full_input_filesthesis.m to transform the hexadecimal binary data files, which were created from the FPGA and data acquisition system into the uncalibrated acceleration data that full_input_filesthesis.m can read.

```
function [accel,OK]=hex_accel(test_no,nwf);
%26/1/2000
if exist('nwf')==1
   if ~isempty(nwf)
      no_weighting_filter_to_be_applied=nwf;
   end
end

tic;
OK=1;
fid=fopen(test_no,'r')%opens the desired file
[A,count]=fread(fid,inf,'char');%reads the opened file
[r,c]=size(A);
s=fclose(fid);%closes the file that was opened
M=r/84;
Mdiff=M-fix(M);
if Mdiff~=0
   warning('Data file you want to process is corrupt.')
   disp('The name of the corrupt file is:')
   disp(test_no)
   OK=0
   return
end
B=reshape(A,84,M)';
[r,c]=size(B);
if B(:,1)==255 & B(:,84)==0
   C(:,1:c-2)=B(:,2:c-1);
else
   warning('Data file you want to process is corrupt.')
   disp('The name of the corrupt file is:')
   disp(test_no)
   OK=0
   return
end
[r,c]=size(C);
for j=1:c/2
   k=2*j;
   if j==1
      D(:,j)=C(:,k-1)+256*C(:,k);
   else
      D(:,j)=256*C(:,k-1)+C(:,k);
   end
end
[r,c]=size(D);
for i=1:c/2
```

```
   j=2*i;
   if i==1
      accel(:,i)=D(:,i);
      %accel(:,i+1)=((D(:,j)./D(:,j+1))-0.5)/0.125;
      accel(:,i+1)=((D(:,j+1)./D(:,j))-0.5)/0.125;
   else
      %accel(:,i+1)=((D(:,j)./D(:,j+1))-0.5)/0.125;
      accel(:,i+1)=((D(:,j+1)./D(:,j))-0.5)/0.125;
   end
end

[accel]=fill_FPGA_gaps(accel);

[accel]=clear_shock_accel(accel);

[accel]=clip_acceleration(accel);

[accel]=is_smooth(accel);

if exist('nwf')==0
   [accel]=iso2631_weighted_filter_Wa(accel);
end

OK
[r,c]=size(accel);
file2accel_time=toc;
disp(char(7));%as 7 is the ascii character for a beep
disp('time taken to covert binary data to acceleration vector of')
disp(test_no)
file2accel_time;
sec=rem(file2accel_time,60);
min=floor(file2accel_time/60);
hour=floor(min/60);
min=rem(min,60);
hour
min
sec
```

fill_FPGA_gapsthesis.m:

This function is called up by hex_accelthesis.m to interpolate data that was missed by the laptop when the information was sent by the FPGA.

```
function [accel]=fill_FPGA_gaps(accel,bob);
bob=accel;
%this is to allow for the fact that the FPGA could have forgotten to
collect some of the
%data so this fills it in with the previously so that it does not
interfer with the calculations
[R,C]=size(accel);
count=R;
numb=1;
ini=accel(1,1);
error_telly=1;
for n=ini:2047
   if n<accel(numb,1)
      backlog=accel(numb:end,:);
      accel(numb,2:end)=accel(numb-1,2:end);
      accel(numb,1)=accel(numb-1,1)+1;
      accel(numb+1:end+1,:)=backlog;
      count=count+1;
      line_error(error_telly)=numb;
      error_telly=error_telly+1;
   end
   numb=numb+1;
end
m=1;
while numb<count
   for n=0:2047
      if n<accel(numb,1)
         backlog=accel(numb:end,:);
         accel(numb,2:end)=accel(numb-1,2:end);
         accel(numb,1)=accel(numb-1,1)+1;
         accel(numb+1:end+1,:)=backlog;
         count=count+1;
         line_error(error_telly)=numb;
         error_telly=error_telly+1;
      end
      numb=numb+1;
      if numb>=count
         break
      end
   end
end

if error_telly-1>0
   %disp(char(7));%as 7 is the ascii character for a beep
   eva=sprintf('There were ''%d'' instances of the FPGA missing data in
this file. \n But this has been accounted for and corrected in this
program.',error_telly-1);
```

```
    warning(eva);
    disp('The errors occurred on the following lines:')
    line_error
end
```

clear_shock_accelthesis.m:

This function is called up by hex_accelthesis.m to eliminate any shock loads outside the measurement scale that the accelerometers have experienced. These shock loads are recorded as infinite. They are replaced by interpolating the surrounding data points to replace these stray measurements.

```
function [accel]=clear_shock_accel(accel);

[i,j]=find(accel==inf);
[I,J]=find(accel==-inf);

if exist('i')==1
   if ~isempty(i)
      n=length(i);
   else
      n=0;
   end
end

if exist('I')==1
   if ~isempty(I)
      m=length(I);
   else
      m=0;
   end
end

if n+m>0
   %disp(char(7));%as 7 is the ascii character for a beep
   eva=sprintf('There were ''%d'' instances of the accelerometers
receiving shock \n loading in this file. \n This has lead to an
inaccurate reading at that point. \n But this has been accounted for
and corrected in this program.',n+m);
   warning(eva);
   disp('These error occurred in the following locations:')
   i,j
end
m=1;
for k=1:n
   o=i(k);
   p=j(k);
   if sign(accel(o-1,p))==-1 %| sign(accel(o-1,p)-accel(o-2,p))==-1
      if j(k)~=0 & i(k)~=0
         II(m)=i(k);
         JJ(m)=j(k);
         i(k)=inf;
         j(k)=inf;
         accel(II(m),JJ(m))=accel(II(m)-1,JJ(m));%-4;%
         m=m+1;
```

```
      end
   end
end
m=1;
for k=1:n
   flag=0;
   if i(k)~=inf
      ii(m)=i(k);
      flag=1;
   end
   if j(k)~=inf & flag==1
      jj(m)=j(k);
      flag=2;
   end
   if flag==2
      m=m+1;
   end
end
clear i,clear j;
if exist('ii')==1
   m=length(ii);
   if ~isempty(ii)
      for s=1:m
         accel(ii(s),jj(s))=accel(ii(s)-1,jj(s));%4;%
      end
   end
end

if exist('I')==1
   m=length(I);
   if ~isempty(I)
      for s=1:m
         accel(I(s),J(s))=accel(I(s)-1,J(s));%-4;%
      end
   end
end
if exist('II')==1
   m=length(II);
   if ~isempty(II)
      for s=1:m
         accel(II(s),JJ(s))=accel(II(s)-1,JJ(s));%-4;%
      end
   end
end
```

clip_accelerationthesis.m:

This function is called up by hex_accelthesis.m to eliminate any measurements outside the possible accuracy of the accelerometers, and then replace it with the maximum possible value.

```
function [accel]=clip_acceleration(accel);

[i,j]=find(accel(:,2:end)>3.95);
[I,J]=find(accel(:,2:end)<-3.95);

if exist('i')==1
   if ~isempty(i)
      accel(i,j+1)=3.95;
      n=length(i);
   else
      n=0;
   end
end
if exist('I')==1
   if ~isempty(I)
      accel(I,J+1)=-3.95;
      m=length(I);
   else
      m=0;
   end
end

if n+m>0
   %disp(char(7));%as 7 is the ascii character for a beep
   eva=sprintf('There were ''%d'' instances of accelerations outside
the measurable \n range of these accelerometers in this file. \n This
has lead to the clipping of the acceleration over +/-4G.',n+m);
   warning(eva);
   disp('Clipping occurred in the following locations:')
   i,j
end
```

is_smoothesis.m:

This function is called up by hex_accelthesis.m and is used to apply a zero-phase forward and reverse butterworth 3rd order low pass filter with a cutoff at 500Hz.

```
function [accel]=is_smooth(accel);

order   = 3;                                          % filter order
(attenuation of 20dB=1 order)
fc      = 0.95;                                          % half of
Nyquist frequency
[b,a]   = butter(order,fc);                          % design of
butterworth filter required
accel(:,2:end)=filtfilt(b,a,accel(:,2:end));%this filters the rocoa
from the butterworth design above
cutoff=500*fc;
fc=fc*100;

eva=sprintf('this file has now undergone Zero-phase forward and reverse
digital \n butterworth filtering, with a order of roll-off ''%d'' and
the cut-off \n frequency is at ''%d'' percent or ''%d''Hz of the
nyquist frequency of ''%d''Hz.',order,fc,cutoff,500);
%disp(char(7));%as 7 is the ascii character for a beep
warning(eva);
```

iso2631_weighted_filter_Wa.m:

This function is called up by hex_accelthesis.m as a specially weighted band pass filter.

```
function [accel]=iso2631_weighted_filter_Wa(accel);
%ISO2631_weigthed_filter_Wa   This function is strictly for use
%  only when data is to be compared to the ISO vibrational standards
%  and strictly only to be used on acceleration data of the following
%  forms:
%
%  Ride Comfort                            = Pass-band filter
%
%  Passenger Comfort in Seated position    = Pass-band filter
%
%  Passenger Comfort in Standing Position = Pass-band filter
%
f=[   0, 0.001,0.005,0.6,   1];
m=[   0,  0.08,    1,  1, 0.7];%the above two lines describe the
%weighting required by the filter.
z2=[   0,   0.000001,0.005,0.6,   1];
z1=[   0,  6*10^-100,    1,  1, 0.7];

[u,v,f,m,p,o]=linear_logrithmic_interpolation(f,m,z1,z2);

b = fir2(1000,u,v);
[h,w] = freqz(b,1,2^20);
figure(1),loglog(f*100,m,w/pi*100,abs(h));
%figure(2),loglog(f*100,m,u*100,v,w/pi*100,abs(h));
title('Frequency weighting, W_a');
xlabel('Frequency (Hz)');
ylabel('Weighting');
grid;
zoom;
orient tall;

%[r,c]=size(accel);
%for i=2:c
%  Zi=ones(max(length(1),length(b))-1,1)*accel(1,c);
%  Zf=ones(max(length(1),length(b))-1,1)*accel(end,c);
%  % filter, reverse data, filter again, and reverse data again
%  [y,Zf]=filter(b,1,accel(:,c),Zi);
%  y=y(length(y):-1:1);
%  [y,Zf]=filter(b,1,y,Zi);
%  accel(:,c)=y(length(y):-1:1);
%end

[accel(:,2:end)]=qfiltfilt(b,1,accel(:,2:end));
```

iso2631_weighted_filter_Wb.m:

This function is called up by hex_accelthesis.m as a specially weighted band pass filter.

```
function [accel]=iso2631_weighted_filter_Wb(accel);
%ISO2631_weigthed_filter_Wb   This function is strictly for use
%  only when data is to be compared to the ISO vibrational standards
%  and strictly only to be used on acceleration data of the following
%  forms:
%
%  Ride Comfort                           = Vertical(z) direction on
floor
%
%  Passenger Comfort in Seated position   = Vertical(z) direction on
floor
%                                          Vertical(z) direction on
seat pan
%
%  Passenger Comfort in Standing Position = Vertical(z) direction on
floor
%
f=[   0, 0.001,0.005,0.02,0.05, 0.2,   1];
m=[   0, 0.035,  0.4, 0.4,   1,   1,0.19];%the above two lines describe
the
%weighting required by the filter.

z2=[   0,   0.001,0.005,0.02,0.05, 0.2,   1];
z1=[   0, 6*10^-9,  0.4, 0.4,   1,   1,0.19];

[u,v,f,m,p,o]=linear_logrithmic_interpolation(f,m,z1,z2);

b = fir2(1000,u,v);
[h,w] = freqz(b,1,2^11);

figure(1),loglog(f*100,m,w/pi*100,abs(h));
%figure(2),loglog(f*100,m,u*100,v,w/pi*100,abs(h));
title('Frequency weighting, W_b');
xlabel('Frequency (Hz)');
ylabel('Weighting');
grid;
zoom;
orient tall;

[accel(:,2:end)]=filtfilt(b,1,accel(:,2:end));
```

iso2631_weighted_filter_Wc.m:

This function is called up by hex_accelthesis.m as a specially weighted band pass filter.

```
function [accel]=iso2631_weighted_filter_Wc(accel);
%ISO2631_weigthed_filter_Wc   This function is strictly for use
%  only when data is to be compared to the ISO vibrational standards
%  and strictly only to be used on acceleration data of the following
%  forms:
%
%  Ride Comfort                              = NO TO BE USED
%
%  Passenger Comfort in Seated position   = longitudinal(x) direction
on seat back
%
%  Passenger Comfort in Standing Position = NOT TO BE USED
%
f=[   0, 0.001,0.005,0.09,   1];
m=[   0,  0.09,    1,   1, 0.09];%the above two lines describe the
%weighting required by the filter.
z2=[   0,   0.0001,0.005,0.09,   1];

z1=[   0, 6*10^-9,    1,  1, 0.09];

[u,v,f,m,p,o]=linear_logrithmic_interpolation(f,m,z1,z2);

b = fir2(1000,u,v);
[h,w] = freqz(b,1,2^11);

figure(1),loglog(f*100,m,w/pi*100,abs(h));
%figure(2),loglog(f*100,m,u*100,v,w/pi*100,abs(h));
title('Frequency weighting, W_c');
xlabel('Frequency (Hz)');
ylabel('Weighting');
grid;
zoom;
orient tall;

[accel(:,2:end)]=filtfilt(b,1,accel(:,2:end));
```

iso2631_weighted_filter_Wd.m:

This function is called up by hex_accelthesis.m as a specially weighted band pass filter.

```
function [accel]=iso2631_weighted_filter_Wd(accel);
%ISO2631_weigthed_filter_Wd   This function is strictly for use
%  only when data is to be compared to the ISO vibrational standards
%  and strictly only to be used on acceleration data of the following
%  forms:
%
%  Ride Comfort                              = longitudinal(x) direction
on floor
%                                              lateral(y) direction on
floor
%
%  Passenger Comfort in Seated position   = lateral(y) direction on
seat pan
%
%  Passenger Comfort in Standing Position = longitudinal(x) direction
on floor
%                                              lateral(y) direction on
floor
%
f=[   0, 0.001, 0.004, 0.02,    1];
m=[   0,  0.08,     1,    1, 0.02];%the above two lines describe the
%weighting required by the filter.

z2=[   0,    0.003, 0.004, 0.02,    1];

z1=[   0, 6*10^-24,     1,    1, 0.02];

[u,v,f,m,p,o]=linear_logrithmic_interpolation(f,m,z1,z2);

b = fir2(1000,u,v);
[h,w] = freqz(b,1,2^11);

figure(1),loglog(f*100,m,w/pi*100,abs(h));
%figure(2),loglog(f*100,m,u*100,v,w/pi*100,abs(h));
title('Frequency weighting, W_d');
xlabel('Frequency (Hz)');
ylabel('Weighting');
grid;
zoom;
orient tall;

[accel(:,2:end)]=filtfilt(b,1,accel(:,2:end));
```

full_cal_files2thesis.m:

This function loads all of the calibration files that are needed for the purposes of calibrating the experimentally collected data.

```
function [vec_cal_cell,AOK]=full_cal_files2(k)
AOK=zeros(1,30);
[px1,AOK(1)]=hex_accel('calxp1.dat',1);
if AOK(1)==1
   vec_cal_cell(1,1)={px1};
end
[px2,AOK(2)]=hex_accel('calxp2.dat',1);
if AOK(2)==1
   vec_cal_cell(1,2)={px2};
end
[px3,AOK(3)]=hex_accel('calxp3.dat',1);
if AOK(3)==1
   vec_cal_cell(1,3)={px3};
end
[px4,AOK(4)]=hex_accel('calxp4.dat',1);
if AOK(4)==1
   vec_cal_cell(1,4)={px4};
end
[px5,AOK(5)]=hex_accel('calxp5.dat',1);
if AOK(5)==1
   vec_cal_cell(1,5)={px5};
end
[nx1,AOK(6)]=hex_accel('calxn1.dat',1);
if AOK(6)==1
   vec_cal_cell(1,6)={nx1};
end
[nx2,AOK(7)]=hex_accel('calxn2.dat',1);
if AOK(7)==1
   vec_cal_cell(1,7)={nx2};
end
[nx3,AOK(8)]=hex_accel('calxn3.dat',1);
if AOK(8)==1
   vec_cal_cell(1,8)={nx3};
end
[nx4,AOK(9)]=hex_accel('calxn4.dat',1);
if AOK(9)==1
   vec_cal_cell(1,9)={nx4};
end
[nx5,AOK(10)]=hex_accel('calxn5.dat',1);
if AOK(10)==1
   vec_cal_cell(1,10)={nx5};
end
[py1,AOK(11)]=hex_accel('calyp1.dat',1);
if AOK(11)==1
   vec_cal_cell(1,11)={py1};
end
[py2,AOK(12)]=hex_accel('calyp2.dat',1);
```

```
if AOK(12)==1
   vec_cal_cell(1,12)={py2};
end
[py3,AOK(13)]=hex_accel('calyp3.dat',1);
if AOK(13)==1
   vec_cal_cell(1,13)={py3};
end
[py4,AOK(14)]=hex_accel('calyp4.dat',1);
if AOK(14)==1
   vec_cal_cell(1,14)={py4};
end
[py5,AOK(15)]=hex_accel('calyp5.dat',1);
if AOK(15)==1
   vec_cal_cell(1,15)={py5};
end
[ny1,AOK(16)]=hex_accel('calyn1.dat',1);
if AOK(16)==1
   vec_cal_cell(1,16)={ny1};
end
[ny2,AOK(17)]=hex_accel('calyn2.dat',1);
if AOK(17)==1
   vec_cal_cell(1,17)={ny2};
end
[ny3,AOK(18)]=hex_accel('calyn3.dat',1);
if AOK(18)==1
   vec_cal_cell(1,18)={ny3};
end
[ny4,AOK(19)]=hex_accel('calyn4.dat',1);
if AOK(19)==1
   vec_cal_cell(1,19)={ny4};
end
[ny5,AOK(20)]=hex_accel('calyn5.dat',1);
if AOK(20)==1
   vec_cal_cell(1,20)={ny5};
end
[pz1,AOK(21)]=hex_accel('calnullp1.dat',1);
if AOK(21)==1
   vec_cal_cell(1,21)={pz1};
end
[pz2,AOK(22)]=hex_accel('calnullp2.dat',1);
if AOK(22)==1
   vec_cal_cell(1,22)={pz2};
end
[pz3,AOK(23)]=hex_accel('calnullp3.dat',1);
if AOK(23)==1
   vec_cal_cell(1,23)={pz3};
end
[pz4,AOK(24)]=hex_accel('calnullp4.dat',1);
if AOK(24)==1
   vec_cal_cell(1,24)={pz4};
end
[pz5,AOK(25)]=hex_accel('calnullp5.dat',1);
if AOK(25)==1
   vec_cal_cell(1,25)={pz5};
end
[nz1,AOK(26)]=hex_accel('calnulln1.dat',1);
if AOK(26)==1
```

```
   vec_cal_cell(1,26)={nz1};
end
[nz2,AOK(27)]=hex_accel('calnulln2.dat',1);
if AOK(27)==1
   vec_cal_cell(1,27)={nz2};
end
[nz3,AOK(28)]=hex_accel('calnulln3.dat',1);
if AOK(28)==1
   vec_cal_cell(1,28)={nz3};
end
[nz4,AOK(29)]=hex_accel('calnulln4.dat',1);
if AOK(29)==1
   vec_cal_cell(1,29)={nz4};
end
[nz5,AOK(30)]=hex_accel('calnulln5.dat',1);
if AOK(30)==1
   vec_cal_cell(1,30)={nz5};
end
```

full_G_calibrationthesis.m:

This function is used to calibrate the data, and ensure that the axes of the accelerometers are orthogonal to each other by applying a rotation matrix.

```
function
[R,t_cal_rCELL,testcal,dir_cal_cell,test_cell_3D,stat_cell_3D,stat_cell
_2D,cal_cel]=full_G_calibration(AOK,vec_cal_cell,test_cell,length_vec,k
,NODOF,rotcal)
%load calibration
for n=1:6;%the six columns rep. '+'x, '-'x, '+'y, '-'y, '+'z and '-'z
   o=n*5

ave=(AOK(o)*(sum(vec_cal_cell{1,o}(:,2:end))/length(vec_cal_cell{1,o}(:
,2:end)))+AOK(o-1)*(sum(vec_cal_cell{1,o-
1}(:,2:end))/length(vec_cal_cell{1,o-1}(:,2:end)))+AOK(o-
2)*(sum(vec_cal_cell{1,o-2}(:,2:end))/length(vec_cal_cell{1,o-
2}(:,2:end)))+AOK(o-3)*(sum(vec_cal_cell{1,o-
3}(:,2:end))/length(vec_cal_cell{1,o-3}(:,2:end)))+AOK(o-
4)*(sum(vec_cal_cell{1,o-4}(:,2:end))/length(vec_cal_cell{1,o-
4}(:,2:end))))/sum(AOK(o)+AOK(o-1)+AOK(o-2)+AOK(o-3)+AOK(o-4));
   stat_cell_2D(n,:)=ave;
end;
%for the 3D component
for i=1:5%this if for the five block which are formed
   m=i*6;
   n=i*4
   for j=1:6
      switch j
         case 1
            stat_cell_3D(1,m-5:m)=[stat_cell_2D(1,n-3:n-
2),stat_cell_2D(5,n-1),stat_cell_2D(1,n-1:n),stat_cell_2D(6,n-3)];
         case 2
            stat_cell_3D(2,m-5:m)=[stat_cell_2D(2,n-3:n-
2),stat_cell_2D(6,n-1),stat_cell_2D(2,n-1:n),stat_cell_2D(5,n-3)];
         case 3
            stat_cell_3D(3,m-5:m)=[stat_cell_2D(3,n-3:n-
2),stat_cell_2D(3,n-1),stat_cell_2D(3,n-1:n),stat_cell_2D(3,n-3)];
         case 4
            stat_cell_3D(4,m-5:m)=[stat_cell_2D(4,n-3:n-
2),stat_cell_2D(4,n-1),stat_cell_2D(4,n-1:n),stat_cell_2D(4,n-3)];
         case 5
            stat_cell_3D(5,m-5:m)=[stat_cell_2D(5,n-3:n-
2),stat_cell_2D(2,n-1),stat_cell_2D(5,n-1:n),stat_cell_2D(1,n-3)];
         case 6
            stat_cell_3D(6,m-5:m)=[stat_cell_2D(6,n-3:n-
2),stat_cell_2D(1,n-1),stat_cell_2D(6,n-1:n),stat_cell_2D(2,n-3)];
      end
   end
end

for n=1:3;%where 1= x-axis, 2= y-axis, 3= z-axis
```

```
   o=n*2;
   range=stat_cell_3D(o-1,:)-stat_cell_3D(o,:);%this is the positive of
the axis minus the negative of the axis
   differ=range/2;
   cal_zero=differ+stat_cell_3D(o,:);
   cal_cel(1,n)={range};
   cal_cel(2,n)={differ};
   cal_cel(3,n)={cal_zero};
 end;
 n=1;
 for kk=1:30
    cal{1}(kk)=cal_cel{2,n}(kk);
    n=n+1;
    if n==4
       n=1;
    end
 end
 for kk=1:30
    cal{2}(kk)=cal_cel{3,n}(kk);
    n=n+1;
    if n==4
       n=1;
    end
 end
 for n=1:3
    cal_cel{1,n}=[]
    cal_cel{2,n}=[]
    cal_cel{3,n}=[]
 end
 %for the normalizing of the calibrated data to give readings in G's
%and also for the calculation of the rotation matrix
%for i=1:3%where 1= x-axis, 2= y-axis, 3= z-axis
%for n=1:6%where 1= '+'x-axis, 2= '-'x-axis, 3= '+'y-axis, 4= '-'y-
axis, 5= '+'z-axis, 6= '-'z-axis
for n=1:6;
   switch n;
      case 1;
         i=1;
      case 2;
         i=1;
      case 3;
         i=2;
      case 4;
         i=2;
      case 5;
         i=3;
      case 6;
         i=3;
   end;      %                                {3,i}
   cal_output(n,:)=(stat_cell_3D(n,:)-cal{2})./cal{1};
end;
%calculation of the rotation matrix
%this is only for a level surface to get the block and the axis aligned
%a rotation matrix is still needed for the accelerometers on the
locomotive
%which needs to be performed every time.
%the basic set up is [F]=[R][Y], where [F] is the desired matrix
```

```
%                                       [R] is the rotation matrix
%                                       [Y] is the matrix we currently
have
Y(1,:)=cal_output(1,:);%this is the calibr for the '+'ve X dir
Y(2,:)=cal_output(3,:);%this is the calibr for the '+'ve Y dir
Y(3,:)=cal_output(5,:);%this is the calibr for the '+'ve Z dir
Y=Y';
for tr=1:10
   out{tr}=Y(tr*3-2:tr*3,1:3)
   R(tr)={-eye(3)/out{tr}}
end

%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
for i=1:k
   test_cell_3D{1,i}=[test_cell{1,i}(:,1:4),test_cell{1,i}(:,4:5),test_
cell{1,i}(:,2),test_cell{1,i}(:,6:8),test_cell{1,i}(:,8:9),test_cell{1,
i}(:,6),test_cell{1,i}(:,10:12),test_cell{1,i}(:,12:13),test_cell{1,i}(
:,10),test_cell{1,i}(:,14:16),test_cell{1,i}(:,16:17),test_cell{1,i}(:,
14),test_cell{1,i}(:,18:20),test_cell{1,i}(:,20:21),test_cell{1,i}(:,18
)]
end

for i=1:k
      testcal(i)={(test_cell_3D{1,i}(:,2:end)-
ones(length_vec(1,i),1)*cal{2})./(ones(length_vec(1,i),1)*cal{1})};%thi
s is for the calibration of the data
      for tr=1:10
         t_cal_rCELL{i}(:,tr*3-2:tr*3)=(R{tr}*testcal{i}(:,tr*3-
2:tr*3)')';%this is for the rotation and the product is the G forces
      end
      intermediate=t_cal_rCELL{i};
      t_cal_rCELL{i}(:,1)=test_cell_3D{1,i}(:,1);
      t_cal_rCELL{i}(:,2:31)=intermediate;
      [r,c]=size(t_cal_rCELL{i})
      [t_cal_rCELL{i}(1:r,2:c)]=rm_dc_offset(t_cal_rCELL{i}(1:r,2:c));
end
```

full_velocitythesis.m:

This function is used to integrate the acceleration.

```
function
[acel,vel,old_time]=full_velocity(tc,old_time,t_cal_rCELL,k,length_vec,
test_t)
%VELOCITY take acceleration data the has been calibrated can
%  then be numerically integrated for the velocity.
%
%  Quinton Rowson
%  10/8/1999
%  Data Analysis
%  This Program is part of the task required for my
%
%  Master of Mechanical Engineering by Thesis
for i=1:k;
   [r,c]=size(t_cal_rCELL{1});
   acel{i}(:,1)=t_cal_rCELL{i}(:,1);   %this is the time stamp for each
line
   acel{i}(:,2:c)=9.81*t_cal_rCELL{i}(:,2:c);   %this converts the
acceleration from G forces to SI unit of acceleration of (m/s/s)
   [ra(i),ca(i)]=size(acel{i}(:,2:c));
   cons_a(i,:)=sum(acel{i}(:,2:c))./(length_vec(1,i)*ones(1,c-1));
   acelnr{i}=acel{i}(:,2:c)-ones(ra(i),1)*cons_a(i,:);   %this was to
get rid of the run-off that is evident in the data when it is
integrated.
   vel{i}(:,2:c)=cumtrapz(test_t{1,i},acelnr{i});
   [vel{i}(1:r,2:c)]=rm_linearline(vel{i}(1:r,2:c));
   [vel{i}(1:r,2:c)]=rm_dc_offset(vel{i}(1:r,2:c));
%   [vel{i}]=iso2631_weighted_filter_Wa(vel{i});
   if etime(clock,old_time)>tc
      old_time=clock
   end
   vel{i}(:,1)=t_cal_rCELL{i}(:,1);
end
```

full_displacementthesis.m:

This function is used to integrate the velocity.

```
function
[dis,old_time]=full_displacment(tc,old_time,vel,k,length_vec,test_t)
%DISPLACEMENT takes numerical velocity data and
%  then numerically integrated for the displacement.
%
%  Quinton Rowson
%  10/8/1999
%  Data Analysis
%  This Program is part of the task required for my
%
%  Master of Mechanical Engineering by Thesis



for i=1:k;
   [r,c]=size(vel{i})
   [rv(i),cv(i)]=size(vel{i}(:,2:c));
   cons_v(i,:)=sum(vel{i}(:,2:c))./(length_vec(1,i)*ones(1,c-1));
   velnr{i}=vel{i}(:,2:c)-ones(rv(i),1)*cons_v(i,:);  %this was to get
rid of the run-off that is evident in the data when it is integrated.
   dis{i}(:,2:c)=cumtrapz(test_t{1,i},velnr{i});
   [dis{i}(1:r,2:c)]=rm_quad_offset(dis{i}(1:r,2:c));
   [dis{i}(1:r,2:c)]=rm_linearline(dis{i}(1:r,2:c));
   [dis{i}(1:r,2:c)]=rm_dc_offset(dis{i}(1:r,2:c));
%  [dis{i}(:,2:c)]=iso2631_weighted_filter_Wa(dis{i}(:,2:c));
   if etime(clock,old_time)>tc
      old_time=clock
   end
   dis{i}(:,1)=vel{i}(:,1);
end
```

full_quintons_PDSthesis.m:

This function calculates the Power Spectral Density (PSD).

```
function
[qa_power,qa_freq,qd_power,qd_freq,old_time]=full_Quintons_PDS(tc,old_t
ime,k,acel,s_f,dis);
%QUINTON_PDS is design to take numerical data and create
%  a power density spectrum of the vector. The sampling
%  frequency of the data must also be supplied so that the
%  P.D.S. can be plotted w.r.t. frequency
%
%  Quinton Rowson
%  10/8/1999
%  Data Analysis
%  This Program is part of the task required for my
%
%  Master of Mechanical Engineering by Thesis

for i=1:k;
   for j=1:30;
      qa_l=length(acel{i}(:,j+1));
      qd_l=length(dis{i}(:,j+1));
      qa_q=fft(acel{i}(:,j+1));
      qd_q=fft(dis{i}(:,j+1));
      qa_p=qa_q.*conj(qa_q);
      qd_p=qd_q.*conj(qd_q);
      qa_power{i}(:,j)=qa_p(1:qa_l/2);
      qd_power{i}(:,j)=qd_p(1:qd_l/2);
      qa_freq{i}(:,j)=(0:(qa_l/2-1))*s_f/qa_l;%this is the frequency of
the vibration
      qd_freq{i}(:,j)=(0:(qd_l/2-1))*s_f/qd_l;
      if etime(clock,old_time)>tc
         old_time=clock
      end
   end
end
```

full_jerkthesis.m:

This function calculates the jerk (rate of change of acceleration).

```
function
[filter_rocoa,old_time]=full_jerk(tc,old_time,k,acel,s_p,length_vec,tes
t_t,test_cell_name)
%JERK take numerical acceleration data and filters and
%  differentiates so that the jerk (which is the rate of
%  change of acceleration) can the be analysied.
%
%  NOTE:That the filtering can be changed depending on
%  what sampling rate was used when the data was collected.
%
%  Quinton Rowson
%  10/8/1999
%  Data Analysis
%  This Program is part of the task required for my
%
%  Master of Mechanical Engineering by Thesis

order   = 3;                                          % filter order
(attenuation of 20dB=1 order)
fc      = 0.95;                                          % half of
Nyquist frequency
[b,a]   = butter(order,fc);                          % design of
butterworth filter required
for i=1:k;
   %accel(i)={acel{i}(:,2:end)};%converts the G forces to accelerations

rocoa(i)={(diff(acel{i}(:,2:end))')./(ones(size(diff(acel{i}(:,2:end)))
)'*s_p)};%this calculates the rocoa
   filter_rocoa{i}=filter(b,a,rocoa{i});%this filters the rocoa from
the butterworth design above
   %[h_but,f] = freqz(b,a,fsize);
   %but_mag   = 20 * log10(abs(h_but));
   %but_phi   = 180/pi * angle(h_but);
   l=length_vec(1,i);
   t=test_t{1,i}(1:l-1);
   if etime(clock,old_time)>tc
      old_time=clock
   end
end;
```

full_RMS_normthesis.m:

This function calculates the RMS acceleration, RMS velocity and the RMS displacement.

```
function
[old_time,a_RMS,v_RMS,d_RMS,a_RMS_f_series,v_RMS_f_series,d_RMS_f_serie
s]=full_RMS_norm(tc,old_time,acel,vel,dis,k,length_vec,test_t,s_p,F_uc,
F_uc_i1,F_uc_i2)

for i=1:k;
   clear x;
   x(:,i)=test_t{1,i}';
   T(i)=x(end,i);
   nargin

   if nargin<10
            a_RMS(i,:)=sqrt((s_p/T(i))*sum((acel{i}(:,2:end)).^2));
            v_RMS(i,:)=sqrt((s_p/T(i))*sum(vel{i}(:,2:end).^2));
         d_RMS(i,:)=sqrt((s_p/T(i))*sum(dis{i}(:,2:end).^2));
         nargout=4
            if etime(clock,old_time)>tc
                  old_time=clock
            end
   else
   %  for n=1:3;
   %     a_RMS(i,n)=sqrt((s_p/T(i))*sum((acel{i}(:,2:end)).^2));
   %     v_RMS(i,n)=sqrt((s_p/T(i))*sum(vel{i}(:,2:end).^2));
   %     d_RMS(i,n)=sqrt((s_p/T(i))*sum(dis{i}(:,2:end).^2));
   %     a_RMS_f_series(i,n)=sqrt((s_p/T(i))*sum(F_uc{i,n}(:,1).^2));
   %
v_RMS_f_series(i,n)=sqrt((s_p/T(i))*sum(F_uc_i1{i,n}(:,1).^2));
   %
d_RMS_f_series(i,n)=sqrt((s_p/T(i))*sum(F_uc_i1{i,n}(:,1).^2));
      %           if etime(clock,old_time)>tc
   %        old_time=clock
   %     end
   %  end
      end
end
a_RMS
v_RMS
d_RMS
if nargout>4
%     a_RMS_f_series
%     v_RMS_f_series
%  d_RMS_f_series
end
```